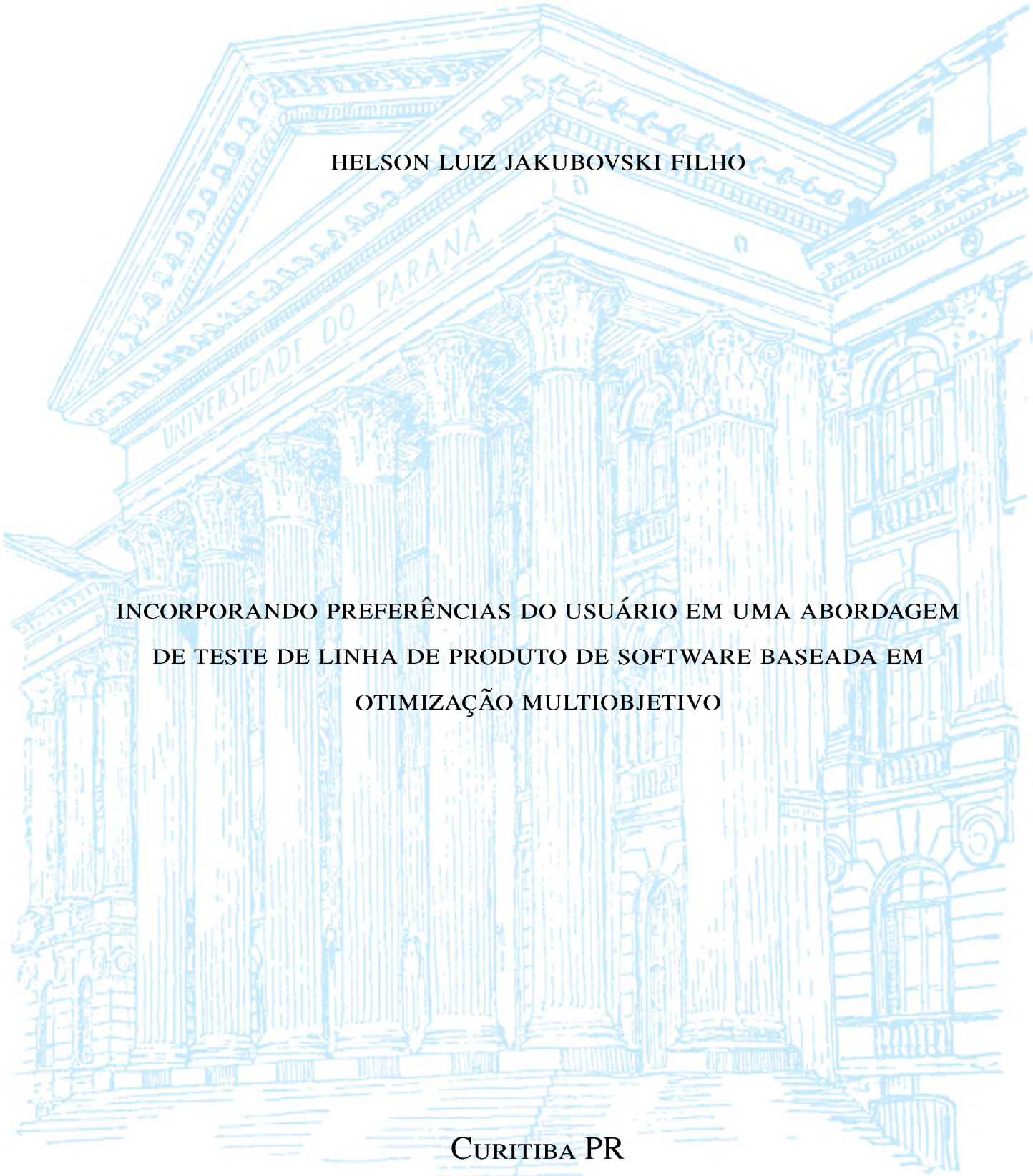
UNIVERSIDADE FEDERAL DO PARANÁ

HELSON LUIZ JAKUBOVSKI FILHO




# INCORPORANDO PREFERÊNCIAS DO USUÁRIO EM UMA ABORDAGEM DE TESTE DE LINHA DE PRODUTO DE SOFTWARE BASEADA EM OTIMIZAÇÃO MULTIOBJETIVO

CURITIBA PR
2018

HELSON LUIZ JAKUBOVSKI FILHO

# INCORPORANDO PREFERÊNCIAS DO USUÁRIO EM UMA ABORDAGEM DE TESTE DE LINHA DE PRODUTO DE SOFTWARE BASEADA EM OTIMIZAÇÃO MULTIOBJETIVO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Profa. Dra. Silvia Regina Vergilio.

CURITIBA PR
2018

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS/UFPR
BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

J25i Jakubovski Filho, Helson Luiz
Incorporando preferências do usuário em uma abordagem de teste de linha de produto de software baseada em otimização multiobjetivo / Helson Luiz Jakubovski Filho. – Curitiba, 2018.
110 p. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2018.

Orientadora: Silvia Regina Vergilio.

1. Linha de produto de software. 2. Teste de software. 3. Algoritmo evolutivo multiobjetivo. 4. Algoritmos baseados em preferência. 5. Hiper-heurística. I. Universidade Federal do Paraná. II. Vergilio, Silvia Regina. III. Título.

CDD: 004.24

Bibliotecária: Romilda Santos - CRB-9/1214

MINISTÉRIO DA EDUCAÇÃO
SETOR CIÊNCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

# TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **HELSON LUIZ JAKUBOVSKI FILHO** intitulada: **Incorporando preferências do usuário em uma abordagem de teste de linha de produto de software baseada em otimização multiobjetivo**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 23 de Março de 2018.

SILVIA REGINA VERGILIO
Presidente da Banca Examinadora (UFPR)

ANDRÉ TAKESHI ENDO
Avaliador Externo (UTFPR)

ROBERTO PEREIRA
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

*Aos meus pais e familiares.*

# Resumo

Algoritmos evolutivos multi-objetivos (*Evolutionary Multi-objective Algorithms* - EMOAs) têm sido aplicados com sucesso para derivar um conjunto de produtos para o teste de variabilidades de Linha de Produto de Software (LPS). Esta tarefa é complexa, impactada por muitos fatores, como o número de produtos a serem testados, os critérios de cobertura a serem satisfeitos e a eficácia em revelar defeitos. Entretanto, implementações com estes algoritmos geralmente produzem muitas soluções que não são interessantes para o testador. Isso acontece porque os algoritmos de busca tradicionais não levam em consideração as preferências do usuário. Para facilitar a seleção das melhores soluções e evitar o esforço gerando soluções não interessantes, este trabalho introduz uma abordagem que aplica algoritmos multi-objetivos evolutivos baseados em preferência (*Preference-Based Evolutionary Multi-objective Algorithm* - PEMOAs) para resolver o problema. Dessa forma são considerados diferentes objetivos sendo o número produtos a serem testados, cobertura *pairwise* e escore de mutação. As preferências do testador são incorporadas antes do processo evolutivo iniciar (a priori) utilizando-se o método de ponto de referência (*Reference Point* - RP). São avaliados dois PEMOAs: R-NSGA-II e r-NSGA-II, usando duas formulações diferentes de objetivos e três tipos de RPs. É apresentada uma abordagem de hiper-heurística (HH) baseada em preferência para resolver esse problema. A abordagem implementa os dois PEMOAs, trabalhando com os métodos de seleção Choice Function, FRR-MAB, e um método aleatório. O objetivo da HH baseada em preferência é realizar a seleção automática de operadores genéticos, guiando a busca pela região de interesse (ROI) do testador. Os experimentos realizados mostram que os PEMOAs apresentam bons resultados, superando o algoritmo tradicional NSGA-II em relação à proximidade das soluções com o RP informado pelo testador e também ao número de soluções geradas dentro e fora da ROI. A utilização da HH baseada em preferência apresentou resultados ainda melhores, superando uma HH tradicional (NSGA-II-HH) encontrada na literatura. A utilização de HHs proporciona mais flexibilidade para o testador por permitir a seleção automática de operadores genéticos. O uso de PEMOAs reduz o esforço do testador na tarefa de selecionar o melhor conjunto de produtos para o teste de LPS.

**Palavras-chave:** Linha de Produto de Software, Teste de Software, Algoritmo Evolutivo Multiobjetivo, Algoritmos baseados em preferência, hiper-heurística.

# Abstract

Evolutionary Multi-Objective Algorithms (EMOAs) have been successfully applied to derive a set of products for variability testing of Software Product Line (SPL). This task is complex, impacted by many factors such as the number of products to be tested, the coverage of some criteria to be satisfied, and the efficacy for revealing faults. However, implementations with these algorithms often produce many solutions that are not interesting to the tester. This is because traditional search algorithms do not take into account user preferences. To facilitate the selection of the best solutions and to avoid the effort generating non-interesting solutions, this work introduces an approach that applies Preference-based Evolutionary Multi-objective Algorithms (PEMOAs) to solve the problem. In this way, different objectives are considered being the number of products to be tested, pairwise coverage and mutation score. The preferences of the tester are incorporated before the evolutionary process (a priori), using the Reference Point (RP) method. Two PMOEAs are evaluated: R-NSGA-II and r-NSGA-II, using two different formulations of objectives and three types of RPs. In addition this work presents a preference-based hyper-heuristic (HH) approach to solve this problem. The approach implements both PMOEAs, working with the selection methods: Choice Function, FRR-MAB, and random. The purpose of a preference-based HH is to perform the automatic selection of genetic operators, guiding the search to a Region of Interest (ROI). The experiments show that the PMOEAs present good results, overcome the traditional NSGA-II algorithm in relation to the proximity of the solutions with RP informed by the tester and also the number of solutions generated inside and outside the ROI. Moreover, the use of the HH based on preference present even better results, better than a traditional HHs (NSGA-II-HH) found in the literature. Besides, the use of HH is more flexible and allow automatic selection of operators. The use of PEMOAs reduces tester's effort to select the best set of products for SPL testing.

**Keywords:** Software Product Line, Software Testing, Multiobjective Evolutionary Algorithm, Preference-Based Algorithms, Hyper-Heuristic.

# Lista de Figuras

# Lista de Tabelas

# LISTA DE ACRÔNIMOS

| | |
|---|---|
| AETG | Automatic Efficient Test Generator |
| AG | Algoritmo Genético |
| AGM | Arcade Game Maker |
| ASF | Achievement Scalarizing Function |
| AVM | Alternating Variable Method |
| CF | Choice Function |
| DE | Distância Euclidiana |
| DM | Decision Maker |
| EA | Evolutionary Algorithm |
| EMOA | Evolutionary Multi-objective Algorithm |
| FaMa | Feature Model Analyser |
| FIFO | First in First Out |
| FIR | Fitness Improvement Rate |
| FM | Feature Model |
| FMTS | Feature Mutation Test Suit |
| FRR | Fitnes-rate-Rank |
| FRR-MAB | Fitness-Rate-Rank Multi-Armed Bandit |
| HH | Hiper-heurística |
| IBEA | Indicator-Based Evolutionary Algorithm |
| LLH | Low Level Heuristic |
| LPS | Linha de Produto de Software |
| MAB | Multi-Armed Bandit |
| MCDM | Multicriteria Decision Making |
| MOP | Multi-objective Optimization Problem |
| NSGA-II | Nondominated Sorting Genetic Algorithm II |
| PEMOA | Preference-based Evolutionary Multi-objective Algorithm |
| PSBSE | Preference and Search-Based Software Engineering |
| R-NSGA-II | Reference Point-based NSGA-II |
| r-NSGA-II | Reference Solution-based NSGA-II |
| R-NSGA-II-HH | Reference Point-based NSGA-II-hiper-heurística |
| RP | Reference Point |
| r-NSGA-II-HH | Reference Solution-based NSGA-II-hiper-heurística |
| ROI | Region Of Interest |
| RPM | Reference Point method |
| SBSE | Search-Based Software Engineering |
| SEI | Software Engineering Institute |
| SPEA2 | Strength Pareto Evolutionary Algorithm 2 |
| SW | Sliding Window |

| | |
|---|---|
| UCB | Upper Confidence Bound |
| VEGA | Vector Evoluation Genetic Algorithm |
| VV&T | Validação, Verificação e Teste |

# Sumário

# 1 INTRODUÇÃO

Uma Linha de Produto de Software (LPS) pode ser definida como um conjunto de produtos que compartilham características comuns e gerenciáveis, e que satisfazem as necessidades de um ramo específico de mercado. O desenvolvimento de um software que utiliza plataformas e customização em massa é resultado da Engenharia de LPS [71, 86]. Essa abordagem tem grande impacto em negócios, organizações, tecnologia e é uma maneira comprovada para desenvolver uma grande quantidade de produtos de software de maneira rápida, com baixo custo, fornecendo um software de alta qualidade [86].

Em engenharia de LPS, uma característica (*feature*) consiste em uma funcionalidade ou atributo do sistema visível ao usuário final, e cada LPS possui um conjunto de características. A combinação de características de uma determinada LPS, ou apenas uma única, é o que forma um determinado produto [71]. As características permitem diferenciar os produtos entre si [86].

Para representar as características e suas relações, modelos de características (*Feature Models* - FMs) podem ser utilizados. Um FM permite a decomposição hierárquica de características sendo estruturado em modelo de árvore de características, e representa a estrutura do conjunto de características de um sistema [71].

Verificar se os produtos gerados através de uma LPS estão de acordo com o desejado consiste em uma importante tarefa exercida dentro da atividade de teste. Assim, para realizar o teste de LPS um produto pode ser considerado como um caso de teste e consiste na derivação customizada das características presentes em uma LPS [18].

Idealmente, segundo Ensan et al. [26], todos os possíveis produtos que podem ser gerados por meio da combinação de características de uma LPS devem ser testados. Entretanto, a quantidade de características pode inviabilizar esta ideia, tendo em vista que à medida que cresce o número de características de uma LPS, o número de possíveis produtos que podem ser gerados cresce exponencialmente [25]. Neste contexto, o teste exaustivo torna-se impraticável tanto em questão de tempo para execução dos testes como em recursos necessários. Esse problema aponta para os principais fatores de interesse quando se fala em teste de LPS: tempo e esforço necessários para testar todos os possíveis produtos [26].

Neste contexto, critérios de teste podem ser utilizados para selecionar e avaliar casos de teste visando a aumentar as possibilidades de revelar defeitos e estabelecer um nível de confiança elevado na correção dos produtos [74]. Na literatura são utilizados critérios como o teste *pairwise* [18, 53, 70, 68, 61] e o teste de mutação [29, 42, 61, 62, 80], com o intuito de selecionar conjuntos de casos de teste para o teste de LPS a partir do FM.

Diferentes conjuntos de produtos que satisfazem os critérios de teste mencionados podem existir. Dessa forma, o desenvolvimento de estratégias para a seleção dos melhores conjuntos de produtos (casos de teste) que garantam a qualidade da LPS torna-se um fator relevante [26]. A seleção dos melhores conjuntos é um problema de busca que pode envolver diferentes fatores, por exemplo, tamanho mínimo do conjunto de casos de teste e maior cobertura para determinado critério. Esse problema, pode ser considerado como um problema da área denominada Engenharia de Software Baseada em Busca (*Search-Based Software Engineering*

- SBSE) [38], que visa à utilização de algoritmos de otimização em problemas complexos da Engenharia de Software.

Levando em consideração que problemas dessa natureza geralmente buscam a otimização de mais de um objetivo, e muitas vezes, acabam sendo conflitantes, a utilização de algoritmos evolutivos multiobjetivos (*Evolutionary Multi-objective Algorithms* - EMOAs) é considerada uma boa opção de solução [16]. Os EMOAs trabalham simultaneamente com um conjunto de possíveis soluções (população), dessa forma, buscam determinar um conjunto de soluções (não-dominadas) que melhor representem o *trade-off* entre os objetivos [12].

No trabalho de Matnei e Vergilio [61] são utilizados três EMOAs para solucionar o problema de teste de LPS descrito anteriormente, sendo eles o NSGA-II (*Non-dominated Sorting Genetic Algorithm-II*) [20], SPEA2 (*Strength Pareto Evolutionary Algorithm* 2) [94] e o IBEA (*Indicator-Based Evolutionary Algorithm*) [78]. Os autores propõem a utilização de três objetivos a serem otimizados, cobertura *pairwise*, tamanho do conjunto de produtos e escore de mutação. Segundo os autores, uma vantagem da abordagem é oferecer ao testador um grande conjunto de boas soluções, com um número reduzido de produtos e altos valores de cobertura. Os autores destacam que de maneira geral o algoritmo NSGA-II obtém os melhores resultados.

Mais recentemente, foram propostos trabalhos que exploram o uso de hiper-heurísticas para resolver este problema [31, 32, 81]. Uma Hiper-Heurística (HH) é uma metodologia para automatizar o projeto e configuração de uma heurística [9] e pode ser usada para determinar automaticamente qual operador de busca (Heurística de baixo nível ou *Low-level heuristic* - LLH) deve ser aplicado no processo de otimização em um dado momento. O uso de HHs para derivar produtos para o teste de LPS oferece abordagens que utilizam EMOAs mais flexíveis e genéricas, obtendo ainda, melhores resultados do que os algoritmos tradicionais [81]. Além disso, é mais fácil usar a abordagem baseada em HH, uma vez que o ajuste de parâmetros e a escolha dos melhores operadores de busca para o problema é uma tarefa difícil para o testador, que geralmente não é especialista na área de otimização.

No trabalho de Strickler et al. [81], os algoritmos NSGA-II, SPEA2 e IBEA foram avaliados para o problema descrito, e os melhores resultados foram obtidos com o algoritmo NSGA-II. Dado este fato, os autores propõem uma HH de seleção de operadores de busca, fazendo uso do algoritmo NSGA-II e de dois métodos de seleção, o método FRR-MAB (*Fitness Rate based Multi-Armed Bandit*) e a seleção aleatória de LLHs. Os resultados mostram que a abordagem baseada em HH apresentou resultados melhores em relação ao algoritmo NSGA-II. Além disso, o método FRR-MAB foi melhor para a grande maioria das instâncias em relação ao método aleatório.

Já no trabalho de Ferreira et al. [31] foi aplicada uma abordagem de HH com o algoritmo MOEA/D-DRA sendo avaliado o desempenho de diferentes métodos UCB (*Upper Confidence Bound*). Todos os métodos apresentaram comportamento semelhante e, novamente, a abordagem baseada em HH superou os algoritmos tradicionais. Estendendo seu trabalho, Ferreira et al. [32] compararam abordagens HH usando MOEA/D-DRA e NSGA-II (o algoritmo que obteve o melhor desempenho em [81]). Novamente, a HH trabalhando com NSGA-II apresentou os melhores resultados.

No entanto, uma limitação comum para o uso prático de tais abordagens permanece. Isso ocorre porque os EMOAs produzem um grande número de soluções (não-dominadas), e muitas vezes, não interessantes do ponto de vista do testador. O testador precisa analisar muitas soluções e, no final, apenas uma delas será usada. Esta limitação é encontrada para este e muitos outros problemas da área de SBSE.

Para lidar com essa limitação, surgiu um novo campo, denominado *Preference and Search-Based Software Engineering* (PSBSE) [33]. O PSBSE é um sub-campo de pesquisa

de SBSE dedicado à aplicação de algoritmos de busca baseados em preferência para resolver problemas de engenharia de software. Um algoritmo baseado em preferências é aquele que incorpora as preferências do usuário (ou *Decision Maker* - DM), intuição, emoção ou aspectos psicológicos no processo de otimização [83].

Existem diferentes maneiras de incorporar as preferências do usuário [8, 33]: antes (a priori), durante (interativamente) ou após (a posteriori) o processo evolutivo. A melhor maneira depende de vários aspectos relacionados à personalidade do usuário, ao contexto, às características do problema, e assim por diante. Por exemplo, considere o problema de teste de LPS. Em muitos casos há um nível de cobertura para o sistema em teste ou um determinado orçamento de custo que são conhecidos a priori. Desta forma, é adequada uma abordagem a priori em vez de uma interativa que requer muito esforço do testador e pode gerar fadiga. Uma maneira de incorporar as preferências do usuário a priori consiste na utilização do método de ponto de referência (*Reference Point* - RP). Para este problema, o testador está interessado apenas em um subconjunto de soluções da fronteira de Pareto que representam uma parte preferida da região ideal de Pareto do ponto de vista do usuário, sendo chamada Região de Interesse (ROI). Então, a preferência do testador orienta o PEMOA durante o processo de busca, o que resulta em um maior número de soluções na ROI, perto do RP. Isso facilita a tarefa de escolher a solução final a ser usada.

Recentemente Ferreira et al. [33] realizaram um mapeamento da área de PSBSE. Dentre os trabalhos encontrados, podem ser mencionados os mais relacionados. O trabalho de Kalboussi et al. [47] que aplica o algoritmo r-NSGA-II (Reference Solution-based NSGA-II) [77] para teste de software de sistemas multiagentes (SMA) e os trabalhos de Yamany et al. [91, 90] que objetivam a configuração de LPS, incorporando as preferências do usuário de forma interativa no processo de evolução. No entanto, o mapeamento mostra que poucos trabalham abordam EMOAs baseados em preferências (PEMOAs), e não relata qualquer trabalho que use abordagens baseadas em HH com PEMOAs.

## 1.1 JUSTIFICATIVA

Diante do contexto apresentado, este trabalho é justificado através das seguintes motivações:

- O critério análise de mutantes é um dos mais eficazes e eficientes em termos de número de defeitos revelados, entretanto um dos mais custosos devido ao número de casos de teste necessários. Assim, uma abordagem que gere automaticamente casos de teste para satisfazer esse critério proporciona a redução de custo e esforço de teste. Os conjuntos de dados gerados devem satisfazer a diferentes fatores tais como possuir um tamanho reduzido, obter altos valores de escore e cobertura de outros critérios de teste, tais como o *pairwise*. Dessa forma, derivar produtos para o teste de variabilidade de LPSs pode ser considerado como um problema multiobjetivo e abordagens multiobjetivo são as mais promissoras.

- Na literatura são encontradas poucas abordagens multiobjetivo que implementam a preferência do usuário durante o processo de teste. Além disso, não foi encontrado um trabalho que inclua a preferência do usuário no teste de LPS. Considerar as preferências do usuário reduz o número de soluções não interessantes geradas e facilita o processo de tomada de decisão da soluções que melhor satisfazem os critérios de teste, a serem utilizadas na prática.

- Trabalhos recentes utilizando hiper-heurística têm apresentado bons resultados para o problema em questão. Porém, não foram encontradas abordagens relacionando hiper-heurística com preferências do usuário. O uso de uma abordagem baseada em HH proporciona uma maior flexibilidade em relação ao uso de EMOAs tradicionais, pois facilita a escolha de operadores de busca e também parâmetros para tais operadores. Dessa forma, o uso de HH proporciona reduzir o esforço do testador.

- A utilização de um método que utiliza interações com o usuário durante o processo de otimização tem a desvantagem de poder causar a fadiga do usuário. Portanto, os algoritmos baseados em preferência guiados por um ponto de referência para o objetivo são bastante adequados para o problema de satisfação dos critérios de teste em LPS. Visto que soluções com baixa cobertura não são interessantes, ou seja, não é difícil definir o ponto de referência, e estes algoritmos não requerem inúmeras avaliações.

- Incorporar preferência do usuário em abordagens multiobjetivo e também baseadas em HH é um tópico de pesquisa ainda não explorado. Os resultados dessa investigação podem ser utilizados em diferentes problemas de SBSE.

## 1.2 OBJETIVOS

Dado o contexto e motivações apresentados anteriormente, o objetivo principal deste trabalho é integrar as preferências do usuário durante o processo de otimização, visando encontrar os melhores produtos para satisfazer o teste de LPS de acordo com as necessidades estabelecidas pelo testador. Para tanto, consideram-se fatores relevantes para o testador como custo relacionado ao número de produtos decorrentes do teste e cobertura de critérios de teste.

Para atingir este objetivo, foram utilizados os PEMOAs já utilizados na literatura de SBSE r-NSGA-II (*Reference Solution-based N:SGA-II*) [77] e R-NSGA-II (*Reference Point-based NSGA-II*) [21]. Três fatores impactaram na escolha destes algoritmos: i) algoritmos baseados no NSGA-II, considerado o melhor algoritmo tradicional aplicado ao problema [61, 81]; ii) consideram a interação com o testador de maneira a priori; e iii) expressam a preferência do DM considerando o método de ponto de referência. Em conjunto com os PEMOAs, utilizou-se uma hiper-heurística de seleção baseada em preferência, proporcionando uma maior flexibilidade e facilidade para o testador estabelecer o processo de busca. Para tanto, três métodos de seleção foram considerados: aleatório, FRR-MAB [54], *Choice Function* (CF) [19]. Assim, deseja-se comparar o desempenho dos PEMOAs descritos anteriormente, considerando diferentes ROIs e objetivos.

## 1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está dividido da seguinte maneira: o Capítulo 2 trata dos principais conceitos relacionados ao teste de LPS. O Capítulo 3 apresenta os principais conceitos relacionados à otimização multiobjetivo, sendo descrito o EMOA NSGA-II. Além disso, ainda são apresentados os principais conceitos relacionados a algoritmos baseados em preferência, sendo descritos os dois algoritmos utilizados neste trabalho: R-NSGA-II e r-NSGA-II. Por fim, este capítulo descreve os principais conceitos relacionados a hiper-heurísticas, sendo descritos os métodos de seleção de LLHs utilizados. O Capítulo 4 apresenta os trabalhos relacionados. Sendo descritos aqueles que abordam o teste de LPS baseado em busca, otimização baseada em preferência em SBSE e também a utilização de hiper-heurísticas em SBSE. O Capítulo 5 descreve a abordagem proposta

e os aspectos de implementação. O Capítulo 6 apresenta a descrição do estudo experimental realizado bem como os resultados obtidos e a discussão destes. O Capítulo 7 apresenta as conclusões obtidas, trabalhos futuros e contribuição deste trabalho. O trabalho contém quatro apêndices. No Apêndice A estão as tabelas contendo os parâmetros utilizados por cada algoritmo. Nos Apêndices B, C e D são apresentadas tabelas de resultados e algumas análises referentes ao experimento descrito no Capítulo 6.

# 2 TESTE DE LINHA DE PRODUTO DE SOFTWARE

Este capítulo tem como objetivo principal apresentar os principais conceitos relacionados ao teste de Linha de Produto de Software. A Seção 2.1 descreve conceitos sobre Linha de Produto de Software (LPS) de forma geral, bem como os aspectos relacionados ao seu desenvolvimento. A Seção 2.2 contém os principais conceitos relacionados ao teste de software. Já na Seção 2.3 é apresentado o contexto de teste de Linha de Produto de Software. Por fim, na Seção 2.4, tem-se as considerações finais sobre este capítulo.

## 2.1 LINHA DE PRODUTO DE SOFTWARE

Uma Linha de Produto de Software (LPS) pode ser definida como um conjunto de produtos que compartilham caraterísticas comuns e gerenciáveis a fim de satisfazerem às necessidades específicas de um determinado ramo de mercado ou missão [11, 86]. O conceito de Linha de Produto de Software consiste na ideia principal de realizar o desenvolvimento de uma família de produtos de software que atenda a um domínio específico de mercado em vez de produzir somente um único produto de software.

A engenharia de LPS apresenta grande impacto em negócios, organizações, e tecnologia, e é uma maneira bem estabelecida para desenvolver uma grande quantidade de produtos de software reutilizáveis de forma rápida e com custos reduzidos, e ao mesmo tempo, fornecendo um software de alta qualidade. Tais características, influenciam diretamente na redução do tempo de desenvolvimento e do esforço de manutenção devido à reutilização de elementos que já foram testados e validados [11].

### 2.1.1 Atividades Essenciais

O processo de desenvolvimento de uma LPS envolve três atividades básicas e necessárias segundo Clements e Northrop [11], sendo elas: desenvolvimento do núcleo de artefatos (*core assets*), desenvolvimento dos Produtos e gerenciamento da LPS.

O desenvolvimento do núcleo de artefatos, pode ser chamado de engenharia de domínio, ou desenvolvimento para o reúso, e tem como objetivo desenvolver o domínio principal, que envolve essencialmente o desenvolvimento da arquitetura e os componentes reutilizáveis. A etapa de desenvolvimento de produtos, pode ser chamada de engenharia de aplicação, ou desenvolvimento com reúso, e busca realizar a geração dos produtos específicos utilizando o núcleo de artefatos e os requisitos específicos do cliente. Por fim, a atividade de gerenciamento da LPS consiste em realizar a manutenção e possível evolução da LPS. Nesta atividade é importante o controle das atividades já desenvolvidas e o planejamento de novas funcionalidades ou melhorias dos artefatos já construídos [11].

Essas três atividades estão intrinsecamente relacionadas de tal maneira que a alteração em uma delas pode consecutivamente resultar em impactos nas demais. A Figura 2.1 apresenta a interação entre as atividades de uma LPS. Nesta figura, os círculos da imagem representam a interatividade das atividades bem como suas interligações.

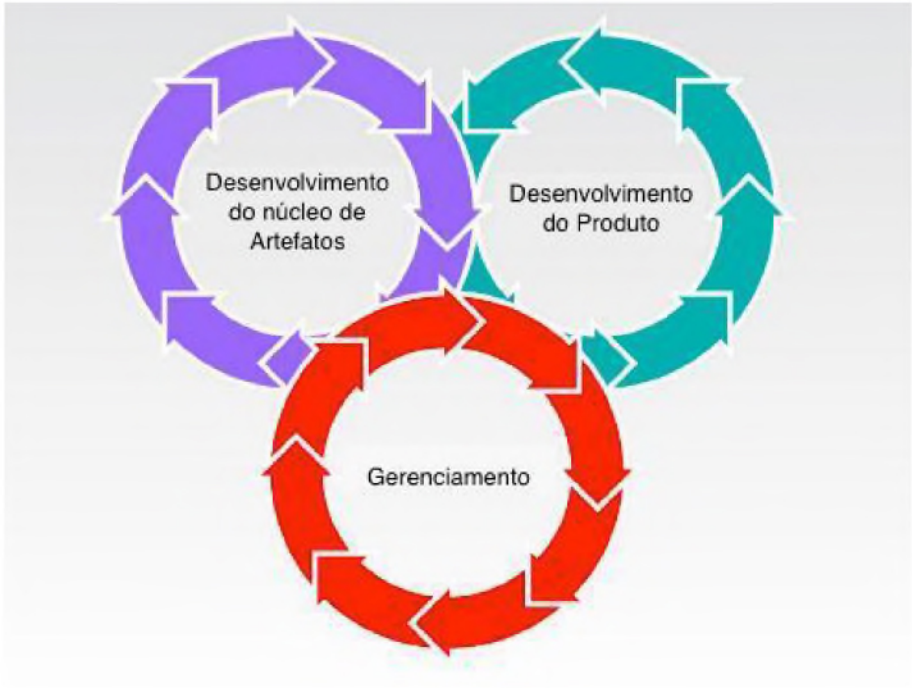



Figura 2.1: Atividades essenciais de uma LPS (adaptada de [44]).

Uma LPS pode ser representada por meio de diagramas de características, que representam o núcleo de artefatos para a confecção de possíveis produtos. A subseção seguinte descreve tais modelos.

### 2.1.2 O Modelo de Características

Na engenharia de LPS, uma funcionalidade ou atributo do sistema que seja visível para o usuário final é denominada característica (*feature*). As características podem ser representadas por meio de um modelo de características (*Feature Model* – FM) [71]. Este modelo define as combinações válidas de características em um domínio. O modelo de características é visualmente representado por uma estrutura em árvore, onde os nós representam as características e as arestas os relacionamentos entre elas. Tal modelo é empregado para demonstrar como as características são utilizadas para a construção de produtos, sendo estes caracterizados pelas *features* que contêm. Um modelo de características deve ser capaz de representar [71]:

- Características mandatórias: Esta deve ser incluída em todos os produtos em que a característica pai aparecer.
- Características opcionais: Esta pode opcionalmente ser incluída em produtos em que a característica pai aparecer.
- Características alternativas: Um conjunto de características é definido como alternativo se apenas uma delas pode ser selecionada quando a característica pai é parte do produto.

- Relação OU: Uma relação entre um conjunto de características é definido como uma relação OU se uma ou mais características podem ser incluídas quando a característica pai é parte do produto.

De maneira geral, um diagrama de características é composto por quatro elementos principais: características, relações, cardinalidade e restrições. Uma característica pode ser uma raiz (*root*), uma característica solitária (*solitary*) ou agrupada (*grouped*). Além disso, as características ainda possuem os atributos nome e domínio. As relações podem ser do tipo binária (*binary*) ou relações agrupadas (*set*). Cardinalidades são aplicadas a características solitárias ou em relações agrupadas. Por fim, as restrições podem ser de dois tipos [28]:

- Inclusão: Essa restrição implica que a escolha de uma características resulta obrigatoriamente a escolha da outra. Dessa forma, se em uma LPS existir uma relação de inclusão de uma característica $x$ para outra $y$, significa que os produtos dessa LPS que contiverem $x$ obrigatoriamente devem conter $y$.

- Exclusão: Essa restrição implica que a escolha de uma característica resulta na exclusão de outra. Dessa forma, se em uma LPS existir uma relação de exclusão de uma característica $x$ para outra $y$, significa que os produtos dessa LPS que contiverem $x$ não podem conter $y$;

Em LPS o conceito de variabilidade representa um papel fundamental, pois é o que possibilita diferenciar os produtos produzidos por uma determinada LPS. A partir da variabilidade, é possível entender cada sistema individualmente, sendo que, a engenharia de LPS olha para a linha de produto como um todo, tentando identificar as variações entre os sistemas. A variabilidade está presente em todos os ciclos de desenvolvimento, desde a análise de requisitos até a definição da arquitetura, componentes, codificação e testes [86].

Alguns elementos devem ser compreendidos para que as variabilidades sejam representadas, sendo eles, pontos de variação e variantes. Um ponto de variação determina os pontos nos quais a variação irá ocorrer, dessa forma, descrevem as características entre os produtos que compõem uma LPS. Cada ponto de variação possui um conjunto de variantes, a variação ocorre por meio da escolha de uma ou mais variantes que podem estar associadas ao ponto. Dessa forma, cada variante representa uma forma de variação, ou seja, representam os possíveis valores que um ponto de variação pode assumir [86].

Na Figura 2.2 é apresentado um exemplo de FM para a LPS *Arcade Game Maker* (AGM), proposta pelo *Software Engineering Institute* (SEI) [44]. As características obrigatórias são representadas pela notação de um círculo preenchido enquanto as características opcionais são representadas por um círculo não preenchido. Características alternativas são representadas por um traço que une duas ou mais características, por exemplo, a característica *rules* que possui como alternativas as opções *brickles*, *pong* e *bowling*. A cardinalidade [1..1] significa que somente é possível escolher uma das 3 características; *action* representa um ponto de variação onde *movement* e *collision* são suas variantes.

## 2.2 TESTE DE SOFTWARE

O processo de construção de software é uma tarefa que, dependendo das características e dimensões do sistema que será desenvolvido, pode se tornar extremamente complexa. Nesse contexto, diversos problemas podem acabar surgindo, podendo até mesmo resultar em um sistema diferente do esperado [22].

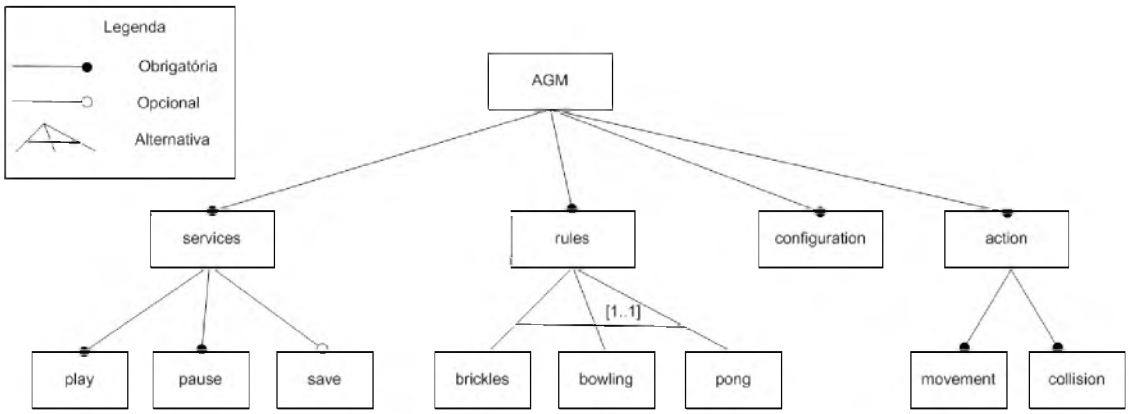



Figura 2.2: FM da LPS AGM (adaptada de [44]).

As atividades de Validação, Verificação e Teste (VV&T) consistem em processos com o objetivo de identificar erros antes do software ser liberado para utilização. Além disso, garantem que tanto o modelo pelo qual o software está sendo construído quanto o produto estejam seguindo as especificações estabelecidas. Nesse contexto, as atividades de VV&T possuem uma função crítica para assegurar a qualidade do software representando a última revisão de especificações, projetos e códigos [22].

Segundo Pressman [73], existem características que incentivam a atividade de teste ser bem planejada e cautelosa, sendo elas, o crescente destaque do software como elemento de sistemas e os altos custos associados às falhas. O autor ainda afirma que é comum encontrar organizações que gastam pelo menos 40% do esforço total de um projeto com a atividade de teste.

### 2.2.1 Conceitos da Atividade de Teste

Para Myers e Sandler [65], o teste consiste em um processo que visa a analisar um programa a fim de encontrar possíveis defeitos. Além disso, ainda tem como objetivo demonstrar que o programa atende às especificações iniciais e trabalha conforme o esperado, fornecendo ainda indicações de confiabilidade e qualidade do software. Durante o teste, uma série de casos de teste deve ser criada com a intenção de revelar problemas no sistema. Por essa característica, o processo de teste é visto como um processo destrutivo em relação à Engenharia de Software.

Outros aspectos referentes ao teste de software, são apontados por Myers e Sandler [65], sendo eles:

- A atividade de teste consiste em executar um programa com o objetivo de revelar defeitos;
- Um bom caso de teste é aquele que apresenta grandes chances de detectar defeitos ainda não descobertos;
- O teste é considerado bem sucedido se consegue revelar um defeito ainda não descoberto.

A seguir são apresentados os principais conceitos relacionados à atividade de teste, adotados pela IEEE [43]:

- Defeito (*Fault*): passo, processo ou definição de dados incorretos;
- Engano (*Mistake*): ação humana capaz de produzir um defeito;

- Erro (*Error*): estado inconsistente ou inesperado de um programa em sua execução;
- Falha (*Failure*): produto de um resultado incorreto produzido pela execução do programa com relação ao resultado esperado.

Além dos conceitos citados, outras definições devem ser levadas em consideração, devido a importância durante a atividade de teste. O domínio de entrada de um programa $P$ é definido como o conjunto de todos os valores possíveis que podem ser empregados para execução de $P$. Um dado de teste, consiste em um elemento que pertence ao domínio de entrada do programa. Já um caso de teste, é um par, composto pelo dado de teste e o resultado esperado pela execução do programa com o determinado dado de teste [73]. Por fim, um conjunto de teste representa o conjunto de todos os casos de teste utilizados no teste [22].

Existem diferentes técnicas de teste, sendo que cada uma é diferenciada pela fonte que utiliza para definir os dados de teste. Dentre as principais destacam-se a técnica funcional, técnica estrutural e a técnica baseada em defeitos. Essas técnicas são abordadas na próxima subseção.

### 2.2.2 Técnicas de Teste

Segundo Myers e Sandler [65] as técnicas de teste podem ser diferenciadas, essencialmente, pela fonte utilizada na definição dos requisitos da atividade de teste. Além disso, cada técnica de teste objetiva explorar um tipo específico de defeito, estruturando requisitos de teste de tal maneira que os valores específicos do domínio de entrada do programa sejam bem definidos com o intuito de exercitá-los.

O teste funcional visa a projetar casos de teste para um programa onde são fornecidas as entradas e avaliadas as saídas verificando se estas estão em conformidade com os objetivos especificados, ou seja, o programa em teste é considerado uma caixa preta. Essa técnica possui como principal característica desconsiderar os detalhes de implementação avaliando o software apenas pelo ponto de vista das funcionalidades disponíveis ao usuário [65].

Uma técnica funcional bastante conhecida é o teste combinatorial, que se baseia na teoria de que a interação entre parâmetros de um software pode revelar defeitos. Para tanto, este tipo de teste faz uso de um *array* de cobertura de casos de teste, gerado através de um mecanismo de amostragem [67]. Segundo Kuhn et al. [51], geralmente, defeitos são revelados apenas pela combinação de dois parâmetros não usuais, justificando a popularidade de técnicas como o *pairwise*.

No teste *pairwise* todas as possibilidades de pares de combinações entre valores de parâmetros são cobertas por pelo menos um caso de teste. A utilização da técnica requer menos esforço que um teste exaustivo, e ainda, pode identificar potenciais defeitos simples, resultantes da interação entre parâmetros, com um conjunto relativamente pequeno de casos de teste [51]. No trabalho de Tai e Lai [82] é descrita uma exemplificação do conceito de *pairwise* levando em consideração um sistema com os valores e parâmetros descritos a seguir:

- Parâmetro $A$ possui os valores $A_1$ e $A_2$
- Parâmetro $B$ possui os valores $B_1$ e $B_2$

Para os parâmetros $A$ e $B$, o conjunto de teste *pairwise* ($A_1$, $B_1$), ($A_1$, $B_2$), ($A_2$, $B_1$), ($A_2$, $B_2$) é o único existente. Esse conjunto leva em consideração todas as combinações dois a dois dos valores dos parâmetros, combrindo dessa maneira, todos os possíveis casos de teste [82]

Na técnica estrutural, para geração dos casos de teste faz-se uso da estrutura interna do programa, necessitando assim, a execução de suas partes ou de componentes elementares. O

objetivo principal da técnica é avaliar os caminhos lógicos do software, estabelecendo casos de teste que sejam capazes de exercitar conjuntos específicos de condições ou ainda laços, tal como, pares de definições e uso de variáveis. Os critérios de teste baseados em fluxo de controle e em fluxo de dados, são exemplos da utilização da técnica estrutural [65].

O teste baseado em defeitos, faz uso de defeitos conhecidos para gerar os requisitos de teste. A motivação principal desta técnica consiste nos possíveis erros que os programadores e projetistas do sistema podem cometer durante o processo de desenvolvimento do software [73]. Destaca-se entre os principais critérios dessa técnica a Análise de Mutantes (ou Teste de Mutação) [23], que consiste em produzir casos de teste específicos para avaliar o programa em teste.

Proposto por DeMillo et al. [23] o critério Análise de Mutantes baseia-se em dois princípios: Hipótese do Programador Competente e Efeito de Acoplamento. O primeiro princípio, assume que programadores competentes escrevem códigos corretos ou bem próximos de estarem corretos. Sendo assim, pode-se afirmar que defeitos são introduzidos nos programas por meio de desvios sintáticos responsáveis por mudarem a semântica do programa, resultando em um possível comportamento incorreto do software. O objetivo da Análise de Mutantes é identificar os defeitos sintáticos mais comuns, para encontrar erros nos programas. O segundo princípio afirma que defeitos complexos são compostos por defeitos menores e mais simples. Desta forma, casos de teste desenvolvidos para identificar defeitos simples também são capazes de revelar defeitos mais complexos.

Para realizar o teste de um programa $P$ utilizando o critério Análise de Mutantes, são inseridas pequenas alterações sintáticas em $P$ produzindo um conjunto de programas mutantes. Tais alterações são realizadas utilizando-se operadores de mutação, que em geral, representam um tipo ou classe de defeitos que podem ser identificados em $P$. O programa $P$ e os diversos programas $P'$ mutantes gerados, devem ser executados utilizando um conjunto de casos de teste. Se o programa mutante $P'$ apresentar um resultado diferente do programa $P$, ele é considerado morto, caso contrário, é considerado vivo. Se o mutante permanece vivo é necessário verificar se $P$ e $P'$ são equivalentes. Para tanto, não deve existir nenhum caso de teste que diferencie a saída produzida entre eles, ou seja, para todos e quaisquer casos de teste os programas resultam sempre o mesmo resultado.

Após a execução dos mutantes, é verificada a adequação dos casos de testes através do escore de mutação. O escore de mutação consiste em um valor real dentro do intervalo [0..1]. Quanto mais próximo de 1, mais adequado é o conjunto de casos de teste, ou seja, maior é a sua capacidade em matar mutantes. A Equação 2.1 apresenta como o escore de mutação pode ser obtido.

$$MS(P,T) = \frac{DM(P,T)}{M(P) - EM(P)} \tag{2.1}$$

Onde:

- $MS(P,T)$: Escore de mutação.
- $DM(P,T)$: Número de mutantes mortos.
- $M(P)$: Número de mutantes gerados.
- $EM(P)$: Número de mutantes equivalentes.

O critério Análise de Mutantes apresenta duas desvantagens. A primeira, deve-se a seu alto custo de execução. O processo de teste acaba gerando uma grande quantidade de programas

mutantes e todos os programas devem ser executados por um determinado conjunto de casos de teste, dessa forma, o tempo computacional para execução do critério se torna alto, mesmo em programas pequenos. A segunda limitação deve-se a existência de mutantes equivalentes. A tarefa de determinar se dois programas computam a mesma função, é considerada uma questão indecidível, necessitando que o testador a realize de maneira manual e isto pode elevar os custos.

## 2.3 TESTE DE LINHA DE PRODUTO DE SOFTWARE

Assim como pode ser observado no desenvolvimento de sistemas tradicionais, o teste de LPS apresenta o objetivo de detectar defeitos nos artefatos produzidos [72]. Além disso, técnicas e métodos de teste convencionais se mostram como alternativas válidas para ajudar no processo de teste de LPS [84]. Contudo, as características peculiares inerentes a uma LPS podem acarretar em determinados problemas e desafios específicos que devem ser tratados.

Segundo Silveira Neto [66], alguns estudos abordam o teste de LPS tanto na fase de engenharia de domínio como na engenharia de aplicação. Na engenharia de domínio algumas atividade de teste estão relacionadas ao desenvolvimento de casos de teste e execução de testes com objetivo de avaliar a qualidade do núcleo de artefatos. As duas principais atividades incluem o desenvolvimento de artefatos de teste que possam ser reutilizados de forma eficiente durante a engenharia de aplicação e a realização de testes no núcleo de artefatos criados durante a engenharia de domínio. Na engenharia de aplicação, são realizados testes adicionais visando a determinar problemas em configurações e assegurar que as variações específicas para um produto estão realmente presentes. Sendo assim, a atividade de teste relaciona-se a seleção e instanciação de artefatos para construir casos de teste específicos.

Como descrito anteriormente, modelos de características são utilizados para representar uma LPS. Em tais diagramas devido a representação hierárquica das características, é possível gerar diferentes produtos, por meio da combinação de características. Dessa forma, um ponto importante está em como tratar as variabilidades, levando em consideração a questão de como diferentes tipos de pontos de variações devem ser testados. Deste modo, torna-se importante a verificação da não existência de relações incorretas entre pontos de variação, ou seja, que características indesejadas não estejam presentes em produtos [25].

Idealmente, segundo Ensan et al. [26], todos os possíveis produtos que podem ser gerados por meio da combinação de características de uma LPS devem ser testados. Entretanto, a quantidade de características pode inviabilizar está ideia, tendo em vista que à medida que cresce o número de características de uma LPS, o número de possíveis produtos que podem ser gerados cresce exponencialmente [25]. Da mesma forma, o tempo para execução do teste acompanha o crescimento exponencial.

Neste contexto, o teste exaustivo torna-se impraticável tanto em questão de tempo para execução dos testes como em recursos necessários para a geração dos casos de teste. Esse problema aponta para os principais fatores de interesse quando se fala em teste de LPS: tempo e esforço necessário para testar todos os possíveis produtos [26].

Levando em consideração as diferentes técnicas para realizar o teste de variabilidades em LPS, frequentemente são encontrados trabalhos que usam o teste combinatorial [18, 53, 70, 68]. A ideia é selecionar as configurações de produtos que cobrem as interações entre as características. Nesse contexto, o teste *pairwise* é o mais utilizado [18]. A ideia é realizar a cobertura de combinações feitas a partir de pares de características.

Recentemente, critérios baseados em defeitos, como a Análise de Mutantes, têm sido empregados para realizar o teste de variabilidades usando o FM [30, 42]. Os estudos demonstram

que esta abordagem é mais eficaz em revelar defeitos, entretanto é mais custosa, em relação à quantidade de casos de teste necessários.

Ambas as técnicas, mutação de variabilidades e teste *pairwise*, são de grande interesse para o desenvolvimento deste trabalho e são descritas nas subseções seguintes.

### 2.3.1 Teste de Mutação em LPS

A abordagem de teste de mutação em LPS foi inicialmente introduzida por Stephenson et al. [80], sendo destacados dois problemas ao se testar uma família de produtos: (i) Quanto as características devem ser testadas antes de serem agrupadas em um produto? (ii) Como saber se um produto corresponde aos seus requisitos e como distinguir diferentes produtos de uma mesma família de produtos?

A fim de resolver o primeiro problema foi proposta a realização de teste de unidade nos artefatos centrais, e apenas realizar teste em variantes quando estas estiverem incorporadas a produtos finalizados. Para a segunda questão, é utilizado o teste de mutação para identificar diferenças superficiais entre instâncias de uma LPS. Para exemplificar, é apresentado um exemplo no qual existe o problema de decidir entre duas características, A e B, para compor um produto. Desta forma o conjunto de produtos desta LPS pode ser dividido entre dois grupos, o primeiro contém a característica A, e o outro a característica B. Dessa forma, os casos de testes produzidos devem ser capazes de mostrar a diferença entre os dois produtos. Por fim, as saídas podem ser comparadas a fim de verificar qual dos produtos mais satisfaz os requisitos especificados [80].

No trabalho de McGregor [62], é apresentada uma abordagem para auxiliar no problema de como determinar a melhor solução para testar uma LPS. Para tanto, é apresentado um modelo de defeitos, que identifica quais defeitos são prováveis de estarem presentes em uma instância de uma LPS. De modo geral, os modelos de defeitos são utilizados pelos testadores para projetar casos de teste efetivos e eficientes, uma vez que são especificados para buscar por possíveis problemas específicos que podem estar presentes. Além disso, os modelos de defeitos ainda podem ser utilizados para determinar o nível de esforço que deverá ser empregue para detectar classes de defeitos específicas.

Em seu trabalho, Henard et al. [42], propõem que em uma expressão *booleana*, que nesse caso representa um FM em teste, sejam inseridas diferentes alterações sintáticas com a finalidade de gerar FMs mutantes. Para tanto, dois operadores de mutação foram propostos com o objetivo de produzir casos de testes dissimilares para revelar defeitos no FM. Já o trabalho de Ferreira et al. [30] apresenta uma abordagem mais completa para geração de mutantes. Primeiramente foram identificadas possíveis classes de defeitos que podem ocorrer em diagramas de características. Com base nessas classes, foram propostos dezesseis operadores de mutação capazes de revelar tais defeitos.

A Figura 2.3 apresenta um exemplo de aplicação de operador de mutação. Nesse caso, o operador de mutação cria uma nova restrição de exclusão, adicionada entre as características *pong* e *configuration*.

Para que o diagrama mutante seja considerado morto é necessário gerar um produto (dado de teste), que seja válido para o programa original e ao mesmo tempo inválido para o mutante, ou vice versa. Um produto é considerado válido se respeita as restrições impostas em um determinado diagrama. Para determinar se um diagrama está valido um analisador de FM deve ser utilizado, o trabalho de Ferreira et al. [30] faz uso do *framework* FaMa [6].

No trabalho de Ferreira [28] é proposto um processo de teste baseado em operadores de mutação de FM. Inicialmente, a partir de um diagrama de características válido, o teste é dividido em três partes:

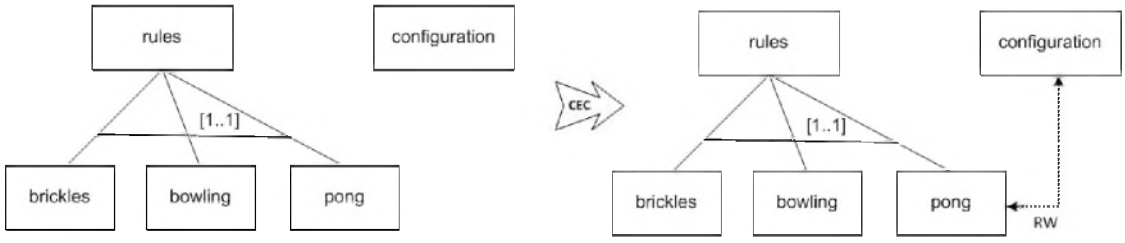



Figura 2.3: Exemplo de mutação no diagrama FM (adaptada de [28]).

- Geração de mutantes: Consiste na aplicação dos operadores de mutação no diagrama de características de entrada. Essa etapa produz um conjunto de diagramas mutantes como saída.
- Geração de dados de teste: Essa etapa tem como saída um conjunto de produtos (dados de teste) válidos para o diagrama de características de entrada ou para algum dos diagramas mutantes gerados.
- Execução dos dados de teste: A partir das duas etapas anteriores, esta consiste em avaliar os produtos gerados, utilizando os mutantes produzidos. Dessa forma, consiste em checar se os produtos fornecidos são válidos para os diagramas mutantes e também para o diagrama de entrada.

Para apoiar ao processo de teste proposto, Ferreira [28] implementou a ferramenta FMTS (*Feature Model Test Suit*) que trabalha em conjunto com o *framework* FaMa. Levando em consideração o exemplo de mutação apresentado na Figura 2.3. O diagrama original é capaz de gerar 7 produtos distintos, a partir da combinação de características (Tabela 2.1). Já o diagrama mutante, é capaz de gerar apenas 6 produtos distintos (Tabela 2.2), isto ocorre devido a restrição de exclusão imposta.

Tabela 2.1: Possíveis produtos que podem ser gerados para o diagrama original do exemplo de mutação.

| | |
|---|---|
| 1 | rules; brickles; configuration; |
| 2 | rules; bowling; configuration; |
| 3 | rules; pong; configuration; |
| 4 | rules; brickles; |
| 5 | rules; bowling; |
| 6 | rules; pong; |
| 7 | configuration; |

Tabela 2.2: Possíveis produtos que podem ser gerados para o diagrama mutante do exemplo de mutação.

| | |
|---|---|
| 1 | rules; brickles; configuration; |
| 2 | rules; bowling; configuration; |
| 3 | rules; brickles; |
| 4 | rules; bowling; |
| 5 | rules; pong; |
| 6 | configuration; |

Considerando cada produto descrito nas duas Tabelas (2.1 e 2.2) como um dado de teste, nota-se que o produto número 3 gerado pelo diagrama original (rules; pong; configuration) é capaz de matar o diagrama mutante da Figura 2.3, tendo em vista que este não consegue derivar tal produto.

Ao final do teste, a ideia é que todos os mutantes que foram gerados sejam mortos por algum caso de teste. Caso o conjunto de dados de teste gerado não satisfaça esse critério, é necessário identificar os diagramas equivalentes e/ou gerar novos dados de teste, visando maximizar o escore de mutação [30]. Um FM mutante é equivalente ao FM sendo testado se ambos derivam o mesmo conjunto de produtos.

### 2.3.2 Teste Combinatorial em LPS

O teste combinatorial é bastante utilizado no contexto de LPS, sendo destacada a grande utilização do teste *pairwise*. De maneira geral, o teste *pairwise* trabalha com a cobertura de combinações feitas a partir de pares de características. Dentre os trabalhos encontrados na literatura, pode-se destacar o de Cohen et al. [18], onde são empregados o teste *pairwise* e vetores de cobertura para realizar a seleção de produtos para o teste de LPS, utilizando o diagrama OVM.

Já no trabalho de Perroin et al. [70], é aplicado o teste *t-wise* com o objetivo de reduzir o número de casos de teste comparado com a abordagem exaustiva, assegurando ainda uma cobertura de teste razoável. O teste *t-wise* consiste em uma metodologia de combinação de $t$ características, ou seja, a quantidade de características a serem combinadas consiste em um número variável. Para solucionar o problema, os autores propõem um conjunto de ferramentas para derivar automaticamente os casos de teste e satisfazer o critério *t-wise*.

O trabalho de Oster et al. [68] realiza a combinação de um algoritmo para a geração de características *pairwise* com o teste baseado em modelo. O objetivo do trabalho é realizar a redução do conjunto de casos de teste necessários para realizar a cobertura de interações entre as características. O trabalho baseia-se na observação de que a maioria dos defeitos são causados através da interação de, pelo menos, duas características.

Lamancha e Usaola [53] também propõem um trabalho para assegurar a cobertura das características pelo teste *pairwise*. A diferença do trabalho consiste na criação de regras que representam os relacionamentos entre as características para a construção de matrizes de pares de características. A partir destas matrizes e fazendo uso de uma versão modificada do algoritmo AETG [17] são gerados os produtos que satisfazem os requisitos de cobertura.

O algoritmo *Automatic Efficient Test Generator* (AETG) é empregado para geração de dados de teste que cobrem todas as combinações validas de $n$ parâmetros. O tamanho de um conjunto de dados de teste gerados pelo AETG cresce de forma logarítmica em relação ao número de parâmetros de teste. Esta característica possibilita que o testador defina modelos de teste utilizando um conjunto reduzido de dados de teste [17].

## 2.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as principais definições e conceitos relacionados ao teste de Linha de Produto de Software. De forma resumida, uma LPS consiste em um conjunto de sistemas que compartilham um conjunto comum e gerenciável de características que tem como objetivo satisfazer os requisitos específicos de um domínio. Uma LPS é composta por um núcleo de artefatos, o qual representa todas as características que são compartilhadas entre todos os possíveis produtos.

A LPS oferece algumas vantagens em relação ao seu uso, como a alta qualidade dos produtos, melhor análise de requisitos e principalmente a redução do custos de produção e manutenção. Entretanto, necessitam de grandes investimentos para implantação apresentando riscos iniciais.

Pelo fato de uma LPS gerar uma família de produtos, é importante assegurar que todos os produtos gerados estejam de acordo com os requisitos estabelecidos. Sendo assim, para assegurar que uma LPS está gerando produtos de qualidade, a realização de testes é de fundamental importância para assegurar que os produtos de software atendam aos requisitos e apresentem boa qualidade. Existem diferentes critérios de teste que podem ser aplicados em LPS, porém, destaca-se a utilização do teste *pairwise* e do teste de mutação de variabilidades.

O critério Análise de Mutantes requer que casos de testes sejam produzidos a fim de matar o conjunto de mutantes gerados. Dessa forma, satisfazer tal critério não é considerada uma tarefa fácil, tendo em vista os diversos fatores envolvidos a serem satisfeitos como a quantidade de produtos, restrições de custo e cobertura de outros critérios, tais como o teste *pairwise*. Nesse contexto, tal problema pode ser modelado como um problema de otimização, mais especificamente, tendo em vista que o fator (objetivo) a ser otimizado não será único, pode ser expresso como um problema de otimização multiobjetivo. Conceitos e algoritmos relacionados à otimização multiobjetivo são descritos no próximo capítulo.

# 3 OTIMIZAÇÃO MULTIOBJETIVO

Neste capítulo são fornecidos os principais conceitos relativos à otimização de problemas multiobjetivos. Na Seção 3.1 são apresentados conceitos referentes aos problemas multiobjetivos. A Seção 3.2 apresenta a estrutura geral de um algoritmo evolutivo e o algoritmo evolutivo multiobjetivo NSGA-II é descrito. Na Seção 3.3 são apresentados os conceitos relativos a algoritmos baseados em preferência, assim como, os PEMOAs utilizados nesse trabalho. Por fim, a Seção 3.4 apresenta os conceitos relativos a hiper-heurísticas.

## 3.1 PROBLEMAS MULTIOBJETIVO

Diversos problemas de otimização do mundo real visam a otimizar um conjunto de objetivos simultaneamente. Porém, em muitos casos, tais objetivos possuem características conflitantes, implicando em uma inexistência de solução única que otimize todos os objetivos ao mesmo tempo. Por exemplo, considere o problema de realização de uma viagem de automóvel, onde se deseja minimizar o tempo de viagem e o consumo de combustível (custo). Tendo em vista que quanto mais se acelera o veículo para reduzir o tempo de viagem mais o gasto de combustível é elevado, é visível que a otimização de um objetivo implica na deterioração de outro.

Os problemas dessa natureza são denominados problemas de otimização multiobjetivo (*Multiobjective Optimization Problems* - MOPs). Nesse contexto, não há apenas uma solução adequada, mas sim um conjunto de boas soluções. Dessa forma, em tais problemas, busca-se otimizar simultaneamente todos os objetivos determinando um conjunto de soluções que representem o melhor *trade-off* (equilíbrio) entre todos os objetivos [15].

De uma formal geral, um problema de otimização multiobjetivo pode ser definido como o problema de minimizar (podendo ser análogo a um problema de maximização) a Equação 3.1 [12].

$$F(x) = (f_1(x), f_2(x), ..., F_k(x)) \tag{3.1}$$

Sujeito às Equações 3.2 e 3.3.

$$G_i(x) \leq 0, \quad \text{para} \quad i = 1, 2, ..., m \tag{3.2}$$

$$H_j(x) = 0, \quad \text{para} \quad j = 1, 2, ..., p \quad x \in \Omega \tag{3.3}$$

onde $k$ representa o número total de funções objetivo, sendo que $f_1(x), f_2(x), ..., f_k(x))$ correspondem às funções objetivo a serem minimizadas, $x$ é um vetor de variáveis de decisão $x = (x_1, x_2, ..., x_n)$ e $\Omega$ corresponde ao conjunto finito de todas as possíveis soluções (espaço de busca) que satisfazem $F(x)$. Além disso, $G_i(x) \leq 0$ e $H_j(x) = 0$ são restrições de desigualdade e igualdade que podem existir em um determinado problema de otimização durante o processo de minimização de $F(x)$ [12].

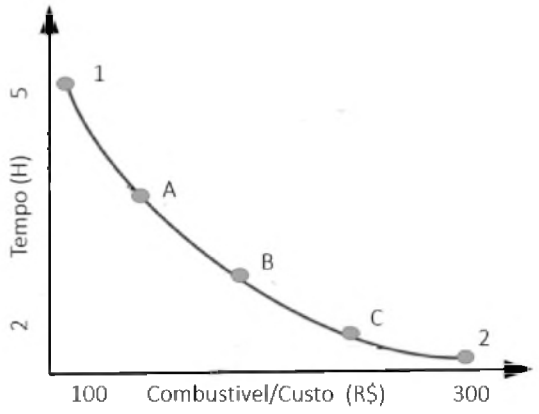



Figura 3.1: Exemplo fronteira de Pareto.

Dentro do espaço de busca ($\Omega$) encontram-se inúmeras soluções factíveis que apresentam bons *trade-offs*, entretanto, é necessário definir quando os objetivos de uma solução são melhores ou piores do que outra solução, ou seja, comparar as soluções encontradas. Para tanto, utiliza-se o conceito de Dominância de Pareto [69]. Considerando duas soluções $x \in \Omega$ e $y \in \Omega$, para um problema de minimização, a solução $x$ domina $y$ se e somente se:

- $x$ não é pior que $y$ em todos os objetivos; e
- $x$ é melhor que $y$ em pelo menos um dos objetivos.

Se não existir nenhuma solução $y$ que domine $x$, então $x$ é dita uma solução não-dominada. Em outras palavras, uma solução é dita não-dominada quando não há nenhuma outra solução admissível que melhore simultaneamente todos os objetivos, ou seja, a melhoria de um objetivo só é alcançada ao custo de prejudicar, pelo menos, um outro objetivo [13]. Através desse conceito, é possível determinar as soluções que apresentem o melhor *trade-off* entre os objetivos de um problema. Essas soluções formam um conjunto de soluções não-dominadas no espaço de busca do problema. O conjunto de soluções não-dominadas para determinado problema é utilizado para formar a Fronteira de Pareto. Dessa forma, a Fronteira de Pareto consiste em um conjunto de soluções candidatas a solucionar um problema multiobjetivo [15].

A Figura 3.1 mostra uma Fronteira de Pareto com base em um esquema de tempo/combustível (custo) para o problema de viagem de um automóvel descrito anteriormente, onde se deseja reduzir o tempo e o custo para realização da viagem. Os pontos 1, A, B, C e 2 representam as soluções não dominadas para o problema e a linha formada pelos mesmos representa a fronteira de Pareto. Nos extremos da fronteira encontram-se os pontos 1 e 2 que representam, respectivamente, as viagens com o menor custo e com o menor tempo. Entre os extremos encontram-se os pontos A, B e C que não apresentam o menor tempo nem o menor custo, mas são soluções que equilibram tais objetivos.

A busca pela solução dos problemas de otimização multiobjetivo pode ser realizada através de diferentes abordagens, sendo uma delas os Algoritmos Evolutivos. Estes algoritmos são considerados uma boa opção para a resolução dos MOPs, pelo fato de adotarem uma população de soluções visando a uma busca que explora eficientemente o espaço de soluções [14]. Os Algoritmos Evolutivos são descritos a seguir.

## 3.2 ALGORITMOS EVOLUTIVOS MULTIOBJETIVOS

Segundo a teoria de Charles Darwin [20], a seleção natural é definida como a preservação de indivíduos mais adaptados ao ambiente. Dessa forma, na natureza tais indivíduos são aqueles que apresentam as maiores chances de sobreviver, competir e reproduzir. Durante o processo de evolução, normalmente, os indivíduos que apresentam características inferiores são eliminados. Tais características são controladas por unidades, denominadas genes, que formam um conjunto chamado cromossomo. Durante as novas gerações, os melhores indivíduos sobrevivem e também repassam suas características genéticas para os descendentes durante o processo de reprodução.

Os Algoritmos Evolutivos (*Evolutionary Algorithms* - EAS) são inspirados na teoria da seleção natural e evolução biológica dos seres vivos, sendo empregados para otimização de processos considerados complexos. Esses algoritmos trabalham com uma população formada por um conjunto de cromossomos. Geralmente, um cromossomo representa uma única solução em um espaço de soluções. Pela utilização de operadores de recombinação (ou cruzamento) e mutação, os quais realizam alterações nos códigos genéticos dos indivíduos, novos elementos são gerados. Caso apresentem melhor código genético, os novos elementos são mantidos, substituindo seus antecessores. Para determinar se um elemento é melhor do que outro, faz-se uso de uma função de avaliação, denominada função de *fitness* [49].

Existem diferentes tipos de algoritmos evolutivos, sendo um deles os Algoritmos Genéticos (AGs). Os passos básicos do funcionamento de um AG são descritos a seguir [49]:

1. Inicialização: Consiste em gerar uma população com soluções candidatas, geralmente confeccionadas de maneira aleatória.

2. Avaliação: Com a população inicial gerada ou uma nova população descendente criada, avalia-se o fitness das soluções candidatas.

3. Seleção: Consiste em selecionar soluções com os melhores valores de fitness, para receberem operações genéticas.

4. Cruzamento: O cruzamento realiza a combinação de partes de duas soluções pais para criar novas soluções denominadas filhos, que possivelmente, apresentarão melhor valor de *fitness*.

5. Mutação: A mutação introduz mudanças aleatórias nas características do indivíduo, o que promove a diversidade genética na população.

6. Atualização: A população descendente criada a partir da seleção, cruzamento e mutação substitui a população pai.

7. Enquanto a condição de parada do algoritmo não for alcançada, os Passos 2-6 devem ser repetidos.

Algoritmos Evolutivos também são utilizados para resolver problemas de otimização multiobjetivo. Tais algoritmos são denominados *Evolutionary Multiobjective Algorithms* (EMOAs). Os EMOAs trabalham simultaneamente com um conjunto de possíveis soluções (população), o que possibilita que sejam encontradas várias soluções não-dominadas, consideradas ótimas, em apenas uma execução do algoritmo [12].

O primeiro EMOA implementando foi o *Vector Evaluation Genetic Algorithm* (VEGA). Esse algoritmo foi introduzido em meados da década de 1980 tendo como objetivo principal

solucionar problemas de aprendizado de máquina. Foi a primeira vez que um algoritmo genético foi utilizado para resolver problemas multiobjetivos e, desde então, uma ampla variedade de algoritmos neste sentido foram propostos na literatura [15].

O algoritmo *Non-dominated Sorting Genetic Algorithm* II (NSGA-II) [20] é um dos mais utilizados no ramo de SBSE [39] e apresentou bons resultados para o problema de LPS [37, 55], incluído o problema de seleção abordado neste trabalho [60, 61]. O algoritmo NSGA-II é descrito na seção seguinte.

### 3.2.1 *Non-dominated Sorting Genetic Algorithm II* (NSGA-II)

O *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [20] é um EMOA baseado em Algoritmo Genético que possui forte estratégia de elitismo, sendo considerado um dos algoritmos mais eficazes para resolução de problemas multiobjetivos. O NSGA-II visa a realizar a classificação das fronteiras fazendo uso de dois procedimentos de ordenação: a ordenação da população em um ranking por não-dominância (*Fast Non-dominated Sorting Algorithm*) e a ordenação individual pela distância de multidão (*Crowding Distance Sorting*), realizando a busca por soluções bem distribuídas no espaço [14, 20]. O pseudocódigo do NSGA-II é apresentado no Algoritmo 1. O algoritmo recebe como entrada o tamanho da população $N$, número de gerações $g$ e as funções que serão otimizadas $f_k(x)$, sendo $k$ o número de objetivos que serão otimizados. A população inicial é gerada com base no tamanho da população recebido como entrada (linha

2). Os indivíduos são ordenados com base em seu *rank* (linhas 3 e 4) e então a população filha é gerada por meio de cruzamento e mutação (linhas 5 a 10).

**Algoritmo 1:** Pseudocódigo do NSGA-II (adaptado de [15]).

```
   Entrada: N, g, f_k(x)
 1 início
 2     Inicializa a população P_0 de tamanho N;
 3     Avalia os valores dos objetivos dos indivíduos de P_0;
 4     Atribui um rank para os indivíduos de P_0;
 5     Gera a população filho Q_0;
 6     início
 7         Seleção por torneio binário;
 8         Cruzamento e mutação;
 9         Calcula os valores dos objetivos dos indivíduos criados;
10     fim
11     para t = 1 até g faça
12         R_t = P_t + Q_t;
13         para cada x ∈ R_t faça
14             Atribui um rank para x;
15         fim
16         Gera as fronteiras de soluções não dominadas (F_t) de acordo com o rank de
            cada solução;
17         Calcula o crowding distance para cada solução de F_t;
18         População P_{t+1} com as soluções das melhores fronteiras F;
19         Gera a população filho Q_{t+1};
20         início
21             Seleção por torneio binário;
22             Cruzamento e mutação;
23             Calcula os valores dos objetivos dos indivíduos criados;
24         fim
25     fim
26 fim
```

A cada geração do algoritmo, os indivíduos das populações pais e filhos obtidos são ordenados e classificados de acordo com a dominância entre as soluções, formando diversas fronteiras (linhas 14 e 16). A primeira fronteira corresponde às soluções não-dominadas de toda a população, a segunda é constituída por todas a soluções que passam a ser não-dominadas, após a retirada das soluções da primeira fronteira, a terceira é composta por soluções que passam a ser não dominadas após a retirada das primeira e segunda fronteiras, e assim consecutivamente até que todas as soluções estejam classificadas em alguma fronteira.

Em cada fronteira uma ordenação é realizada utilizando uma outra medida, denominada distância de multidão (*crowding distance*), que tem como objetivo assegurar a diversidade das soluções (linha 17). O *crowding distance* calcula o quão distante uma solução está em relação aos seus vizinhos da mesma fronteira. Assim, é possível estabelecer uma ordem decrescente que privilegia as soluções mais espalhadas no espaço de busca. As soluções que estão no limite de espaço de busca possuem somente um vizinho, porém, são as mais diversificadas da fronteira, por isso, recebem altos valores de forma a ficarem no topo da ordenação.

As duas ordenações, de fronteiras e por distância de multidão, são utilizadas para determinar quais indivíduos irão sobreviver para a próxima geração. Além disso, também são

utilizadas pelo operador de seleção (linhas 7 e 21). A criação de novos indivíduos é realizada mediante aplicação dos operadores de cruzamento e mutação (linhas 8 e 22).

O processo de funcionamento do NSGA-II é representado na Figura 3.2, onde $P_t$ representa a população pai e $Q_t$ representa a população filho. $F_1$, $F_2$ e $F_3$ são fronteiras de soluções já ordenadas da união de $P_t$ e $Q_t$. Por fim, $P_{t+1}$ representa o conjunto de soluções que serão usadas na próxima geração, levando em consideração o tamanho da população, formada pelas melhores fronteiras ($F_1$ e $F_2$) e as soluções com a melhor distância de multidão da fronteira $F_3$.

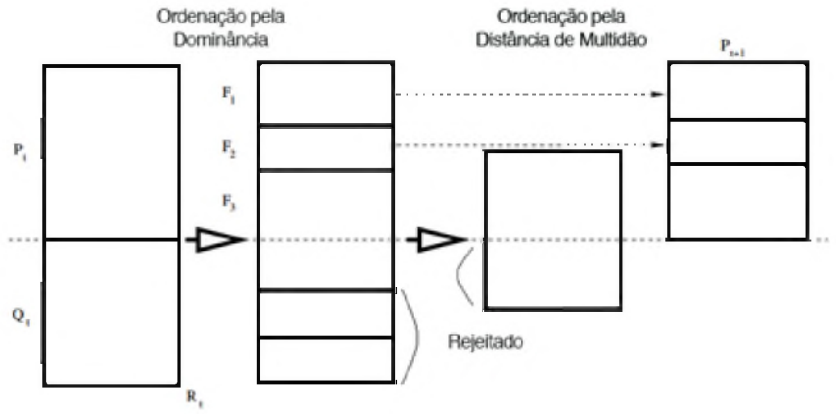



Figura 3.2: Funcionamento do NSGA-II (adaptada de [15]).

O algoritmo NSGA-II gera um conjunto de soluções espalhadas por todo o espaço de busca, formando a frente de Pareto, como descrito anteriormente. Entretanto, por muita vezes o testador esta interessando em uma única solução ou conjunto específicos dessa soluções. Para realizar tal tarefa, torna-se viável a utilização de algoritmos baseados em preferência, que guiam a busca por soluções que expressam especificamente o interesse do testador. Tais algoritmos são descritos nas seções seguintes.

## 3.3 ALGORITMOS BASEADOS EM PREFERÊNCIA

Como descrito anteriormente, um EMOA busca determinar um conjunto de soluções que apresentem o melhor relacionamento entre todos os objetivos do problema, formando assim, a fronteira de Pareto composta por todas as soluções não-dominadas encontradas. No entanto, em aplicações do mundo real, o tomador de decisão (*Decision Maker* - DM) não está interessado em todas as soluções presentes na fronteira de Pareto, pois muitas vezes a decisão final do usuário é uma única solução. O objetivo final de um EMOA é auxiliar o DM a determinar a solução que mais se adeque as suas necessidades. Levando em consideração que os EMOAs fornecem uma grande quantidade de soluções ao DM, determinar a melhor solução pode não ser uma tarefa fácil [4]. A utilização de algoritmos baseados em preferência pode ser uma alternativa para auxiliar na solução deste problema.

O objetivo de um algoritmo baseado em preferência (*Preference-Based Evolutionary Multi-objective Algorithm* - PEMOA) é incorporar as preferências do ser humano no processo de otimização. Para que isso seja possível, o tomador de decisões representa um papel fundamental, sendo ele quem irá interagir com o processo de otimização. Pressupõe-se que o DM é quem possui o melhor entendimento sobre o domínio do problema, e é quem melhor pode expressar as relações de preferência entre as diversas soluções. Dessa forma, estas preferências são utilizadas

para guiar a procura pela região de preferência da fronteira de Pareto, ou seja, pela região de interesse (*Region Of Interest* - ROI) [83].

A ROI consiste em um conjunto de soluções não-dominadas que representam as soluções mais próximas ao interesse do DM. Estas soluções estão co-localizadas dentro da fronteira de Pareto, na região que o DM possui o interesse [1, 21]. A Figura 3.3 apresenta uma fronteira de Pareto baseada na região de interesse do usuário.

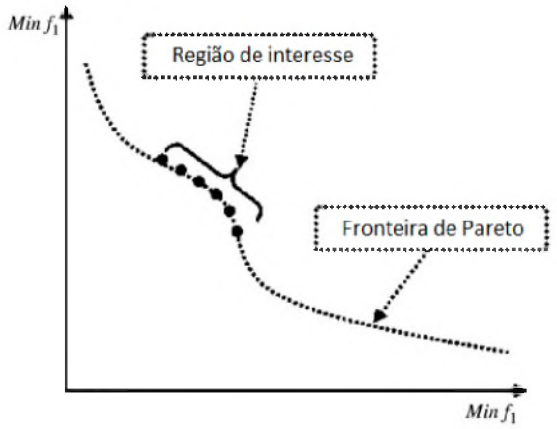



Figura 3.3: Ilustração da região de interesse (adaptada de [77]).

O processo de incorporar a preferência do usuário durante a execução do algoritmo pode ocorrer em diferentes estágios da execução. Said et al. [77] sugerem três momentos distintos:

- *A priori*: As preferências do usuário são expressas antes do processo de busca inciar;
- *A posteriori*: É realizada a interação com usuário somente no final da execução do algoritmo, onde um conjunto de soluções são disponibilizadas para que o tomador de decisão escolha a que melhor se adeque às suas opções;
- Interativa (ou *in-the-loop*): As preferências do usuário são incorporadas durante a execução do algoritmo. Neste caso, o tomador de decisão participa ativamente do processo de busca guiando as soluções de acordo com seus objetivos.

A forma como a informação de preferência é expressa pelo DM pode ocorrer de diferentes maneiras. A maioria dessa formas, são encontradas na literatura referente à tomada de decisão multicritério (*Multicriteria Decision Making* - MCDM) [63], tais como:

- uso de pesos, onde cada peso representa o grau de importância de um determinado objetivo;
- uso de ponto de referência (também chamado de vetor de nível de aspiração), corresponde a pontos no espaço de busca em que o DM gostaria que os objetivos se concentrassem;
- nível de aceitação, o DM informa o valor mínimo de aceitação que cada objetivo pode assumir, para que ainda seja considerado satisfeito;
- uso de *trade-off* entre os objetivos, o DM especifica que o ganho de uma unidade em um objetivo vale a degradação em outros e vice-versa;

- a classificação dos objetivos em diferentes classes: objetivos que devem ser diminuídos (levando em consideração um problema de minimização), objetivo que deve ser reduzido até um nível de aspiração, objetivos que são satisfatórios no momento, objetivos que podem ser aumentados até um nível de aceitação e objetivos livres a alterar a incrementar/decrementar valores.

Dentre as abordagens descritas para expressar as preferências do DM, destacam-se alguns trabalhos que fazem uso do método do ponto de referência (*Reference Point Method* - RPM) [21, 77], e que são de especial interesse para este trabalho.

O RPM foi inicialmente proposto por Wierzbicki [89]. Em um MOP, um ponto de referência $g$ consiste em um vetor de níveis de aspiração, onde são definidos os valores (níveis de aspiração) desejados pelo DM para cada objetivo. Com esta definição, o ponto de referência apresenta-se como uma maneira natural de expressar as preferências do DM. Esse método, realiza a projeção do ponto de referência informado pelo DM sobre a região ótima de Pareto através da minimização de uma função escalar de desempenho (*Achievement Scalarizing Function* - ASF).

A Figura 3.4 apresenta as possibilidades que um ponto de referência pode assumir em relação à fronteira de Pareto, sendo: (A) viável e pertencendo a fronteira de Pareto; (B) viável e não pertencendo a fronteira de Pareto; e (C) inviável, um ponto que não pode ser obtido a partir de qualquer solução encontrada no espaço de busca viável. Nos casos B e C, o DM pode estar interessando em soluções presentes na fronteira de Pareto que mais se aproximam ao ponto de referência.

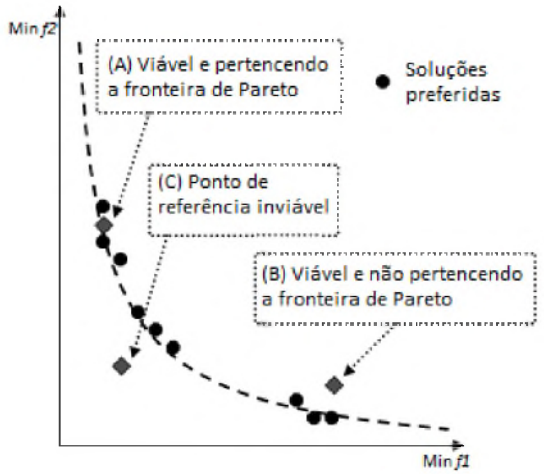



Figura 3.4: Método do ponto de referência (adaptada de [77]).

Para utilizar o método do ponto de referência na prática, é requisitado ao DM que informe o ponto de referência e também um vetor que contém um peso atribuído a cada objetivo. O local informado para o ponto de referência faz com que o procedimento de busca se concentre numa determinada região na fronteira de Pareto. Já o vetor de pesos, possibilita um melhor equilíbrio entre os objetivos do problema, fornecendo uma informação mais detalhada sobre o ponto para onde devem convergir as soluções na fronteira de Pareto [89].

A seguir são descritos dois algoritmos baseados em preferência que serão utilizados neste trabalho.

### 3.3.1 *Reference Point-based NSGA-II* (R-NSGA-II)

No trabalho de Deb et al. [21] é proposta uma abordagem que implementa a preferência do usuário, utilizando o método do ponto de referência, em um algoritmo multiobjetivo. O R-NSGA-II como foi denominado, consiste em uma versão modificada do algoritmo NSGA-II, que tem o objetivo de guiar a busca da ROI de acordo com as preferências do DM. Segundo os autores, o R-NSGA-II é bem aplicável a qualquer forma de fronteira, a grandes números de objetivos e também variáveis. Ainda, possibilita a busca de diferentes ROIs com base em diferentes pontos de referência ao mesmo tempo, em outras palavras, o algoritmo é capaz de encontrar diferentes conjuntos de soluções concorrentemente em relação a diferentes pontos de referência.

O funcionamento do algoritmo é semelhante ao NSGA-II original, porém difere em alguns aspectos. Em primeiro lugar, para aplicar o R-NSGA-II, um DM deve fornecer um ou mais pontos de referência para o algoritmo, caso contrário, ele irá funcionar igual ao NSGA-II. Segundo, a métrica de *crowding distance* é modificada, passando a ser denominada "distância de preferência" devido a representar o quanto as soluções estão próximas dos pontos de referência. A utilização dessa métrica implica em dar uma maior ênfase nas soluções que estão mais próximas aos pontos de referência informados pelo DM. Finalmente, a fim de manter a diversidade de soluções selecionadas próximas aos pontos de referência, o R-NSGA-II aplica uma estratégia de seleção chamada $\epsilon$-*clearing*, que usando um parâmetro denominado $\epsilon$ enfatiza as soluções mais próximas ao ponto de referência.

A métrica de distância de preferência implementada pelo R-NSGA-II funciona da seguinte maneira: (i) é realizado o cálculo da distância euclidiana normalizada (Equação 3.4) de cada solução da fronteira para cada ponto de referência, classificando as soluções em ordem crescente em relação à distância. Assim, a solução que apresentar a menor distância em relação ao ponto de referência, ocupa o primeiro lugar no *rank*, a segunda mais próxima ocupa a segunda posição, e assim por diante; (ii) depois, é realizado o cálculo para todos os pontos de referência, a distância de preferência de uma determinada solução é a classificação mínima que lhe foi atribuída entre todos os diferentes *rankings*. As soluções com os menores valores da distância de preferência são preferidas pelo método de seleção e também na formação da nova população [21].

$$Dist(x,g) = \sqrt{\sum_{i=1}^{M} w_i \left( \frac{f_i(x) - f_i(g)}{f_i^{max} - f_i^{min}} \right)^2}, \tag{3.4}$$

onde $M$ é o número máximo de objetivos, $w_i$ é vetor de pesos, $x$ é a solução considerada, $g$ é ponto de referência especificado, $f_i^{max}$ e $f_i^{min}$ são os valores máximo e mínimo para o $i$-ésimo objetivo. A Figura 3.5 apresenta a classificação de distância de preferência atribuída a cada solução da população, assumindo-se 6 soluções (S1,...,S6) e 2 pontos de referência (R1 e R2). Na Figura 3.5(A), é demonstrado o cálculo da distância para cada solução em relação a cada ponto de referência e o *rank* atribuído. A Figura 3.5(B) apresenta o valor mínimo da classificação de cada solução, que é considerado a distância de preferência da solução.

Para controlar a diversidade de soluções selecionadas próximas aos pontos de referência, R-NSGA-II aplica um procedimento denominado estratégia de seleção $\epsilon$-*clearing*. Para tal processo, primeiramente é escolhida uma solução aleatoriamente no conjunto das soluções não-dominadas. Em seguida, todas as soluções que têm uma soma de diferença normalizada em valores objetivos menores que $\epsilon$ são agrupadas e, em seguida, recebem um alto valor de distância de preferência com o intuito de desencorajá-las a permanecer nas próximas gerações do processo de evolução. Em seguida, é escolhida outra solução do conjunto de soluções não-dominadas

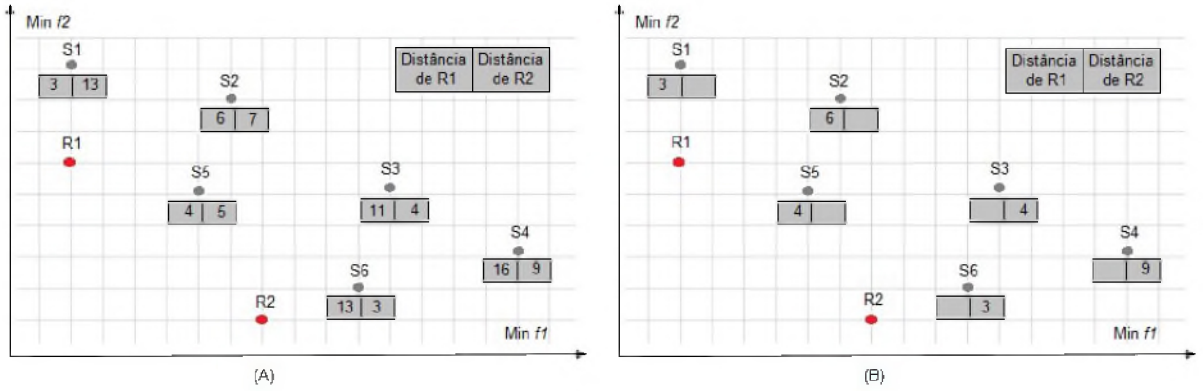



Figura 3.5: Exemplo de atribuição da distância de preferência a cada solução.

(excluindo-se a escolhida anteriormente) e o procedimento é executado novamente. O valor de $\epsilon$ é escolhido de acordo com a aplicação e pode ser diferente para cada objetivo, dessa forma, consiste em uma parâmetro informado pelo DM [21].

A Figura 3.6 apresenta o efeito da variação do valor $\epsilon$ no número de soluções presentes na fronteira. É possível observar que quanto maior for o valor de $\epsilon$ maior a quantidade de soluções e consecutivamente maior o espalhamento das soluções na fronteira.

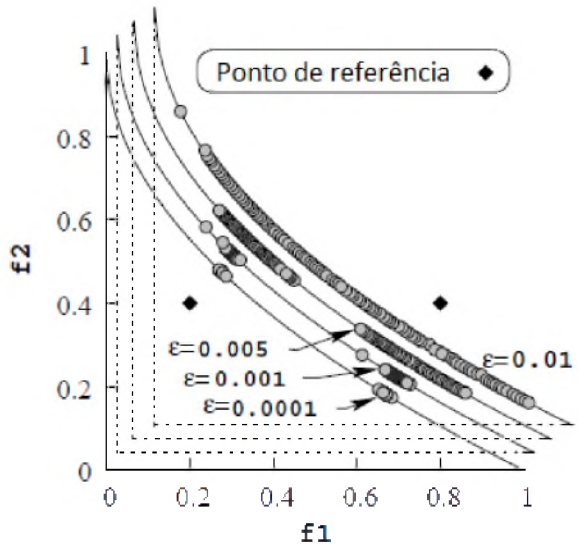



Figura 3.6: Efeito da variação de $\epsilon$ nas soluções (adaptada de [21]).

### 3.3.2 *Reference Solution-based NSGA-II* (r-NSGA-II)

O trabalho de Said et al. [77] também utiliza o conceito de ponto de referência e algoritmo multiobjetivo. Os autores propõem um novo conceito para a dominância de Pareto, denominado dominância baseada em soluções de referência (r-dominância). A principal característica da r-dominância é dar preferência para soluções mais próximas do ponto de referência, fornecido pelo DM, que preservem a ordem induzida pela dominância de Pareto. Para determinar a distância de uma certa solução para o ponto de referência, os autores fazem uso da função de distância euclidiana apresentada anteriormente na Equação 3.4.

O conceito de r-dominância pode ser definido da seguinte forma: assumindo uma população $P$, um vetor de referência $g$ e um vetor de pesos $w$, uma solução $x$ é dita $r$-dominada por uma solução $y$, se uma das seguintes afirmações for verdadeira:

1. $x$ domina $y$ na relação de Pareto;
2. $x$ e $y$ são Pareto equivalentes e $D(x, y, g) < -\delta$, onde $\delta \in [0,1]$ e:

$$D(x, y, g) = \frac{Dist(x, g) - Dist(y, g)}{Dist_{max} - Dist_{min}} \tag{3.5}$$

$$Dist_{max} = Max_{z \in P} Dist(z, g) \tag{3.6}$$

$$Dist_{min} = Min_{z \in P} Dist(z, g) \tag{3.7}$$

onde $\delta$ é denominado limiar de r-dominância.

A ideia principal do conceito de r-dominância é definir uma relação entre soluções equivalentes de Pareto. Assim, a r-dominância tem a capacidade de diferenciar soluções não-dominadas de uma forma parcial com base em um vetor de níveis de aspiração fornecido pelo DM. Este processo, além de tornar a relação de dominância "mais forte" em relação à dominância de Pareto, também possibilita integrar as preferências do DM no processo de busca [77]. Para assegurar que a r-dominância preserva a dominância de Pareto, os autores realizam a demonstração de diferentes provas e definições que podem ser encontradas em [77].

Para verificar a eficiência da nova abordagem, Seid et al. [77] realizaram uma modificação no procedimento de classificação por não-dominância usado pelo algoritmo NSGA-II, substituindo a relação de dominância de Pareto pela nova relação de r-dominância. A ideia da substituição da dominância de Pareto pela r-dominância é classificar uma população de soluções baseadas na preferência do DM, expressas como um ponto de referência, preservando a ordem induzida pela dominância de Pareto.

A Figura 3.7 apresenta um exemplo da classificação de uma população com 16 indivíduos usando o princípio da r-dominância. Para cada indivíduo representado como um círculo preenchido, a esquerda é representada a posição de dominância de Pareto e a direita a r-dominância. Observando a primeira fronteira não-dominada composta pelas soluções (A, B, C e D), nota-se que essa fronteira se divide em duas classes diferentes quando utilizada a r-dominância. As duas são: C1 = {B, C} contendo indivíduos que se encontram mais próximos ao ponto de referência e C2 = {A, D} contendo soluções mais distantes do ponto de referência. Nota-se que as soluções A e D são r-dominadas por uma das soluções B e C. Portanto, a pressão imposta pela relação de r-dominância apresenta-se mais forte do que a dominância de Pareto. Está pressão de seleção é guiada pelas preferências do DM e controlada pelo limiar de r-dominância $\delta$. Além disso, o mesmo fenômeno pode ser observado para as segunda e terceira fronteiras não-dominadas [77].

O parâmetro $\delta$ permite ao DM controlar a pressão de seleção da relação de r-dominância. Quando $\delta = 1$, a relação de r-dominância equivale à dominância de Pareto, já se $\delta = 0$, são enfatizadas somente as soluções não-dominadas que mais se aproximam do ponto de referência [77]. Com essas características, o parâmetro $\delta$ pode ser utilizado para assegurar a diversidade populacional durante o processo evolutivo. Para tanto, Said et al. [77] propõem uma forma de gestão adaptativa do limiar de r-não-dominância $\delta$ durante o processo de evolução, funcionando da seguinte maneira: Assumindo que $n\_ge$ seja o número de gerações que estabelece

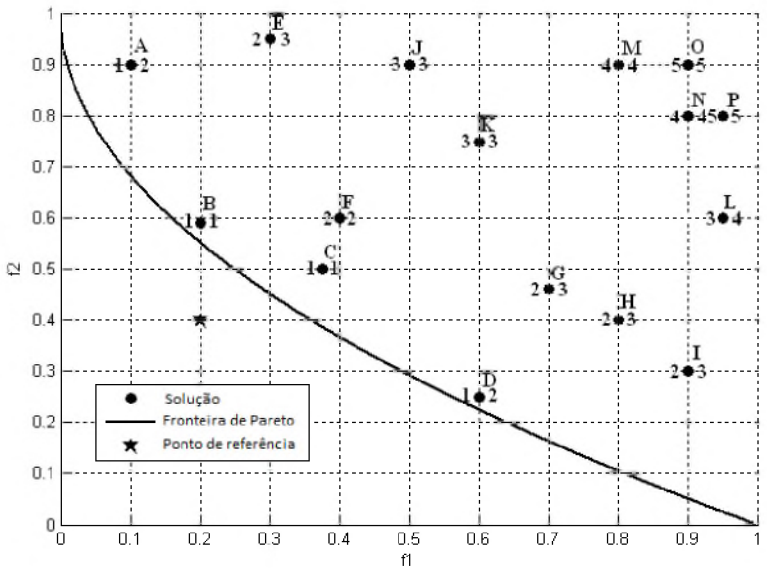



Figura 3.7: Exemplo de solução r-não-dominada (adaptada de [77])

o critério de parada, $i$ seja o índice da geração atual do ciclo evolutivo e $\delta_0$ o valor de $\delta$ fornecido pelo DM; No estágio de inicialização ($i = 0$), o algoritmo r-NSGA-II classifica a população com base na dominância de Pareto ($\delta = 1$). A partir da primeira geração, o valor de $\delta$ é truncado em cada geração, de acordo com a relação "$1 - n\_ge((1 - \delta_0)/n\_ge)$". Sendo assim, na última geração, o valor de $\delta$ corresponde ao valor informado inicialmente pelo DM ($\delta_0$). O objetivo desta gestão adaptativa do parâmetro $\delta$ é orientar a busca gradualmente durante o processo de evolução para determinar a ROI, de forma que seja possível evitar o fenômeno de convergência prematura [77].

De forma geral, o processo do r-NSGA-II é dividido em três passos, sendo eles [77]:

- Passo 1: Solicitar ao DM que forneça o tamanho da população, o critério de parada, a solução de referência (ponto de referência), um vetor de pesos para os objetivos e por fim o valor de $\delta$.
- Passo 2: Executar o algoritmo r-NSGA-II até que o critério de parada seja satisfeito. Essa etapa, consiste na mesma execução do algoritmo NSGA-II (Algoritmo 1), porém, utilizando o conceito de r-dominância.
- Passo 3: Fornecer ao DM o conjunto de soluções obtidas. Se o DM estiver satisfeito com o conjunto de soluções fornecido, então o processo de otimização é encerrado, caso contrário, é verificado se o DM gostaria de atualizar a solução de referência, o vetor de peso, o valor de $\delta$ ou o critério de parada, e em seguida, o algoritmo retorna ao Passo 2.

## 3.4 HIPER-HEURÍSTICAS

Algoritmos multiobjetivos têm sido amplamente utilizados em busca de soluções de diversos problemas na mais variadas áreas de aplicação. Contudo, realizar a implementação de uma solução com esses algoritmos pode não ser uma tarefa fácil, principalmente para quem não possui um bom conhecimento e domínio do campo de otimização. Conforme vão surgindo novos algoritmos e novos problemas, tem-se a dificuldade em determinar qual combinação de algoritmos e operadores seria mais adequada para o problema. Nesse contexto, o conceito de

hiper-heurística surge como uma possível solução para que um processo de busca seja guiado de forma a identificar heurísticas de baixo nível promissoras (*Low Level Heuristics* - LLH) [76], de forma que uma LLH possa assumir a combinação de operadores genéticos, ou até mesmo, meta-heurísticas (como os algoritmos NSGA-II, SPEA2, IBEA, etc).

Segundo Cowling et al. [19] uma hiper-heurística pode ser definida como heurísticas que escolhem heurísticas. No entanto, uma nova definição mais recente foi proposta por Burke et al. [9], como sendo um conjunto de abordagens ou uma metodologia de alto nível com o intuito de proporcionar um projeto ou ajuste automático de métodos heurísticos de modo a resolver problemas computacionalmente difíceis. Adicionalmente, Chakhlevitch e Cowling [10] definem hiper-heurística como uma heurística de alto nível que consiste em três critérios: (i) a hiper-heurística deve gerenciar um conjunto de LLHs; (ii) busca-se um bom método para resolver o problema, mais do que uma boa solução; e (iii) utilizam-se apenas informações limitadas, que são específicas do problema, e que formam a barreira de domínio.

O conceito de barreira de domínio, ou barreira do conhecimento, apresentado na Figura 3.8, tem o objetivo de assegurar que a hiper-heurística possua conhecimento somente de informações não ligadas ao domínio do problema. A hiper-heurística tem acesso apenas a informações como a quantidade de LLHs e a função de avaliação [19]. Dessa forma, a barreira de domínio possibilita que a estratégia de hiper-heurística possa ser usada para qualquer problema, fazendo-se alterações somente nas LLHs implementadas.

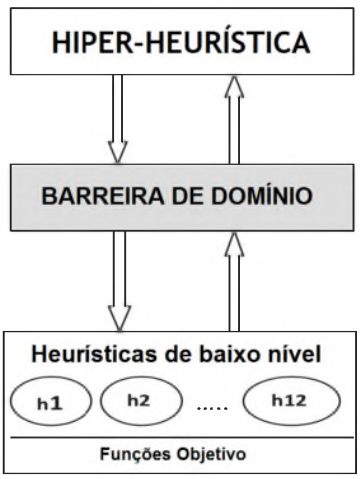



Figura 3.8: Barreira de domínio (adaptada de [9]).

Sendo assim, o uso de hiper-heurística foca-se em realizar o trabalho dentro do espaço de busca de heurísticas ao invés do espaço de busca de soluções, buscando a construção adequada da combinação de algoritmos (ou parte deles) com o intuito de alcançar resultados melhores do que seriam obtidos com a aplicação individual destes [9].

### 3.4.1 Classificação

Uma hiper-heurística pode ser classificada de duas maneiras, segundo a sua natureza e sua capacidade de aprendizado, conforme apresentado na Figura 3.9. Com relação a sua natureza, uma hiper-heurística pode ser classificada como: metodologias de seleção, onde podem ser escolhidas heurísticas existentes, ou hiper-heurística de geração, onde novas heurísticas são geradas através de componentes pré-existentes [9].

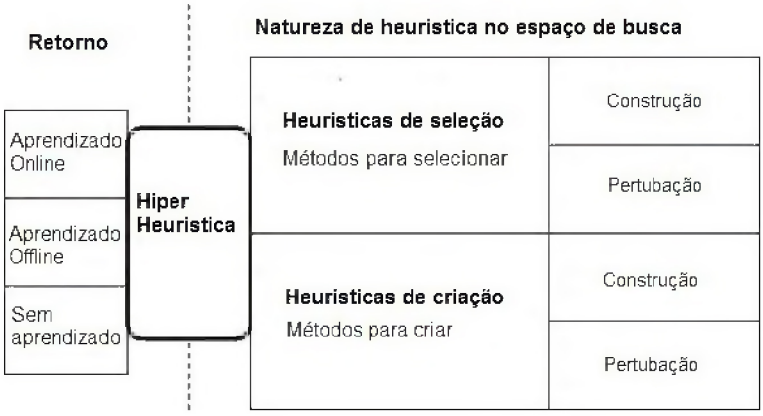



Figura 3.9: Tipos de hiper-heurísticas (adaptada de [9]).

Uma hiper-heurística também pode ser classificada como de natureza construtiva, que trabalha considerando parcialmente soluções candidatas, em que um ou mais elementos não estão disponíveis, e de forma interativa, que cria novos elementos através de elementos anteriores. Ou ainda de natureza perturbativa, que trabalha considerando soluções candidatas completas e as altera através da modificação de um ou mais elementos da solução [9].

Em relação à forma de aprendizado de uma hiper-heurística existem as categorias: aprendizado *on-line*, o processo de aprendizado é realizado enquanto o algoritmo esta resolvendo a instância de um problema; e aprendizado *off-line*, no qual se aplica um método genérico para a resolução a partir de um conjunto de treinamento. O resultado deste treinamento é um algoritmo que potencialmente é mais apto em qualquer instância do problema o qual foi treinado; por fim, caso não haja nenhum tipo de informação, a hiper-heurística é considerada sem aprendizado, como por exemplo, escolha aleatória ou exaustiva de heurísticas [9]. Ainda há casos, onde é utilizada uma metodologia híbrida, mesclando heurísticas construtivas e perturbativas [34], bem como heurísticas de seleção com geração [48]. Este trabalho tem como interesse hiper-heurísticas de seleção *on-line* que serão descritas a seguir.

## 3.4.2 Métodos de Seleção

Uma hiper-heurística de seleção de heurísticas perturbativas geralmente é composta por um par de componentes: (i) método de seleção que visa a selecionar uma LLH; e (ii) método de aceitação que determina se irá aceitar ou não uma nova solução gerada [9]. Os métodos de seleção fazem uso de técnicas para determinar a seleção de uma LLH em um certo momento da busca, podendo ou não fazer uso de aprendizado. Esses métodos fazem uso de otimização adaptativa dinâmica para aprender estratégias, através de uma máquina de aprendizado, sem ter necessidade de supervisão [9, 36], determinando o momento de aplicação de uma LLH no processo de busca.

Existem diferentes técnicas de seleção com ou sem aprendizado. Dentre as técnicas de seleção com aprendizado, destacam-se as baseadas em pontuação (*ranked based*) que basicamente atribuem ou atualizam uma pontuação a uma LLH, aplicada de acordo com regras de pontuação para o desempenho da LLH. Há também técnicas que utilizam o ajuste de memória, o qual avalia uma série histórica de aplicações das LLHS. O procedimento chamado *Credit Assigment* é chamado sempre que são realizadas a atribuição ou atualização da pontuação do desempenho de uma LLH. Para tanto, uma estratégia deve ser aplicada para a seleção de uma LLH, tais

como: *max*, uma LLH com pontuação máxima é selecionada; roleta, faz uso de probabilidades de seleção baseadas na pontuação [9].

Existem diversos métodos que geralmente são utilizados com hiper-heurísticas, dentre ele destacam-se os métodos de seleção *Choice Function* (CF) [19] por sua simplicidade de implementação e melhora nos resultados obtidos, e também o *Fitness Rate Rank Multi-Armed Bandit* (FRR- MAB) [54] devido a seus recentes e promissores resultados na aplicação de hiper-heurísticas.

### *Choice Function* (CF)

O algoritmo de seleção *Choice Function* (CF) teve sua aplicação inicialmente estudada por Cowling et al [19], sendo do tipo *ranked based* para a seleção de uma LLH. O CF mantém uma escore de cada LLH, avaliando, de forma adaptativa, a qualidade de cada aplicação de LLH de modo a escolher a cada ponto de decisão aquela que possua o maior escore de todas para ser aplicada no processo. Para tanto, o CF utiliza uma função matemática (Equação 3.8), onde o elemento $f_1$ representa o quão bom foram os resultados obtidos pela aplicação da LLH $h_i$, o elemento $f_2$ representa se a sequência das LLHs $h_i$ e $h_j$ (executadas anteriormente) é aceitável, e o elemento $f_3$ representa o tempo de espera (em segundos) que a LLH $h_i$ foi mantida inativa. Este elemento possibilita que ocorra uma melhor diversificação na escolha de LLHs. Caso este elemento não seja considerado, somente uma LLH seria escolhida em todo o processo de busca.

$$F(h_i) = \alpha f_1(h_i) + \beta f_2(h_j, h_i) + \delta f_3(h_i) \tag{3.8}$$

onde $\alpha$, $\beta$ e $\delta$ são parâmetros que regulam a equação. Usar um alto valor para $\delta$ aumenta a influência do elemento $f_3$ e permite que as escolhas sejam realizadas de maneira mais uniforme, porém impede que o desempenho de cada heurística seja relevante nas escolhas. Por outro lado, altos valores para $\alpha$ e $\beta$ podem fazer com que os elementos $f_1$ e $f_2$ tenham maior influência e fazer com que as escolhas sejam mais elitistas e priorizem mais uma heurística do que as outras.

Posteriormente, ocorreu uma simplificação para o CF, proposta por Maashi et al. [56]. Foi removido o elemento $\beta f_2(h_j, h_i)$, e o parâmetro $\delta$ de $\delta f_3(h_i)$. A Equação 3.9 descreve a nova CF.

$$F(h) = \alpha f_1(h) + f_2(h) \tag{3.9}$$

Na CF apresentada na Equação 3.9 o componente $f_1$ possui a mesma função do componente $f_1$ da Equação 3.8, refletindo a melhoria recente de $h$, e o componente $f_2$ possui a mesma função do componente $f3$ da Equação 3.8, a qual reflete o tempo desde a última utilização de $h$.

### *Fitness-Rate-Rank based Multi-Armed Bandit* (FRR-MAB)

O problema *Multi-Armed Bandit* (MAB) estuda o dilema entre a Exploração versus Intensificação (*Exploration vs Exploitation* - EvE). A Exploração visa a explorar de forma adequada o conjunto de operadores estabelecidos de modo a dar chance a todos os operadores, uma vez que um operador não selecionado pode se tornar melhor em uma próxima aplicação durante a busca. Já a Intensificação visa a estabelecer com uma maior frequência o operador que obtém bons resultados. Neste contexto, no MAB um jogador tem como objetivo maximizar sua recompensa em um jogo com um conjunto de caça-níqueis. Para isto o jogador deve definir a ordem de execução e quantas vezes deve jogar em cada um dos caça-níqueis. Cada caça-níquel possui uma alavanca ou um braço (*arm*) que é acionado para executar o jogo. Assim, no problema

MAB a escolha de um caça-níquel é representada pela escolha de um braço. Dentre os inúmeros algoritmos propostos para tratar do problema MAB, o algoritmo *Upper Confidence Bound* (UCB) destaca-se na resolução por garantir a otimalidade assintótica no total de recompensas acumuladas [54]. Posteriormente houve outros algoritmos que se inspiraram no UCB, como por exemplo, o *Fitness-Rate-Rank based Multi-Armed Bandit* (FRR-MAB) [54], que vem se destacando por seus resultados focando na tarefa de atribuição de crédito [54].

O FRR-MAB [54] é uma algoritmo de seleção que incorpora alguns atributos do UCB [2] e também possui duas etapas, sendo elas a escolha de LLHs e o método de atribuição de crédito (*Credit Assignment*). O FRR-MAB seleciona automaticamente um determinado operador a ser aplicado, através do aprendizado *on-line*, onde a qualidade de um operador é avaliada pela melhoria do valor de *fitness* obtida [54]. Um mecanismo de decaimento é utilizado para calcular o valor da recompensa final que cada operador ira receber em relação às aplicações realizadas em uma determinada janela de tempo. Posteriormente, o próximo operador a ser selecionado é baseado nesse valor de recompensa. Assim, o FRR-MAB seleciona o melhor operador de acordo com a Equação 3.10.

$$op_t = argmax_{i=1,..,K}\left(FRR_{i,t} + CX\sqrt{\frac{2xln\sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}\right) \tag{3.10}$$

onde o objetivo é selecionar o melhor operador $op$ através da estratégia $argmax$. Para cada ponto de tempo $t$ dentro do conjunto $K$ de operadores, que possua um valor empiricamente estimado $FRR_{i,t}$ em um intervalo que depende do número de vezes $n_{i,t}$ em que o operador (ou LLH) foi aplicado anteriormente. É utilizado um parâmetro de fator escalar $C$ para controlar o *trade-off* entre Exploração e Intensificação.

A etapa de atribuição de crédito consiste em atribuir a recompensa para um determinado operador baseando-se na qualidade de sua aplicação. O primeiro passo no procedimento de atribuição de crédito consiste em realizar o cálculo da qualidade relacionada à aplicação de um determinado operador de modo a mensurar seu impacto na busca, para isso, utiliza-se o método de taxa de melhoria de *fitness* (*Fitness Improvement Rate* - FIR) [54] identificada através da Equação 5.4. Procura-se não utilizar os valores brutos das melhorias de *fitness* causadas pelo uso de determinado operador, pois isso varia de problema para problema e diferentes fases de um problema de otimização. Assim, o uso direto desse valor pode de certa forma deteriorar a eficiência do algoritmo [54].

$$FIR_{i,t} = \frac{pf_{i,t} - cf_{i,t}}{pf_{i,t}} \tag{3.11}$$

onde $i$ consiste em um operador em um tempo $t$, $pf_{i,t}$ representa o valor de *fitness* das soluções pais e $cf_{i,t}$ o valor de *fitness* das soluções filhas [54]. Posteriormente, os valores de FIRs dos operadores recentemente utilizados são guardados em uma janela de tempo (*sliding window* - SW) de tamanho $W$, sendo organizados como uma estrutura de dados de lista e obedecendo a metodologia FIFO (*First In First Out*), ou seja, os itens sempre são adicionados ao final da lista, caso um novo item deva ser adicionado e a quantidade a seja igual a $W$, então o elemento mais antigo da lista deve ser removido para que o novo item possa ser adicionado no fim da mesma. O uso da SW possibilita avaliar um operador sem que ele seja prejudicado pelo seu desempenho em uma fase muito precoce, a qual pode ser irrelevante para o seu desempenho atual. Dessa forma, garante-se que a informação da FIR na SW refira-se a uma situação atual da busca [54].

O Algoritmo 2 mostra o funcionamento da etapa de atribuição de crédito para o FRR-MAB. O algoritmo inicia gerando o vetor auxiliar *Recompensa*, que é um somatório de todos os

valores de FIR existentes na janela de tempo para uma dada LLH, e gerando o vetor $Nt$ (Linha 10). Em seguida, os valores do vetor auxiliar $Recompensa$ são classificados em ordem decrescente (Linha 12) para que cada valor do vetor tenha uma posição (1 a K) na classificação $Rank$. Com os vetores $Recompensa$ e $Rank$, os valores $Decay$ são calculados (Linha 15). Posteriormente, o algoritmo executa o somatório dos valores contidos no vetor $Decay$ para que este seja usado em conjunto com os valores contidos no vetor $Decay$ no cálculo do FRR. Este cálculo é executado para cada LLH (Linha 19). Ao final do algoritmo os vetores FRR e $Nt_i$ são retornados.

**Algoritmo 2:** Pseudocódigo do método de Atribuição de Crédito do FRR-MAB (adaptado de [54]).

```
1  início
2      para cada i ∈ K faça
3          Recompensa_i = 0.0;
4          Nt_i = 0;
5      fim
6      para cada elemento ∈ janela de tempo faça
7          i = elemento.GetOperador();
8          FIR = elemento.GetFIR();
9          Recompensa_i = Recompensa_i+FIR;
10         Nt_i++;
11     fim
12     Ranqueamento das recompensas em ordem decrescente (Recompensa_i);
13     Rank_i = valor de ranking do Recompensa_i;
14     para cada i ∈ K faça
15         Decay_i = D^rank_i X Recompensa_i;
16     fim
17     DecaySum = Σ_{i=1}^{k} Decay;
18     para cada i ∈ K faça
19         FRR_i = Deacy_i / DecaySum;
20     fim
21     retorna FRR;
22 fim
```

O procedimento de seleção de operadores é mostrado no Algoritmo 3. Este algoritmo recebe como parâmetros o vetor FRR (gerado na etapa de atribuição de crédito), $K$ que representa a quantidade de LLHs, o vetor $Nt$ (é o número de aplicações de uma LLH na janela de tempo), e o parâmetro $C$ usado na Equação 3.10. Este algoritmo garante a execução de todas as LLHs por uma rodada, antes de escolher uma LLH pela Equação 3.10, garantindo assim, uma escolha justa.

## 3.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma revisão teórica de conceitos sobre o Problema de Otimização Multiobjetivo e suas formulações. Como descrito anteriormente, para solucionar problemas multiobjetivos, faz-se uso de algoritmos bio-inspirados, mais especificamente, inspirados na teoria da evolução. Sendo assim, foi apresentado o EMOA NSGA-II, a ser utilizado neste trabalho. Além disso, ainda foram apresentados os conceitos e as motivações para utilização de

**Algoritmo 3:** Pseudocódigo do método de seleção do FRR-MAB (adaptado de [54]).

**Entrada:** $FRR$, $C$, $k$, $Nt$

**início**
  **se** *Existem LLHs ainda não selecionadas* **então**
    $op$ = Selecionar aleatoriamente alguma LLH ainda n ao selecionada;
  **fim**
  **senão**
    $op = argmax_{i=1,..,K}\left(FRR_{i,t} + CX\sqrt{\frac{2xln\sum_{j=1}^{K} Nt_j}{Nt_i}}\right)$
  **fim**
  **retorna** op;
**fim**

algoritmos baseados em preferência do usuário, dando-se atenção a dois algoritmos que utilizam ponto de referência para expressar a preferência, sendo eles R-NSGA-II e r-NSGA-II. Além disso, neste capítulo ainda foi apresentado o conceito de hiper-heurística e os métodos de seleção CF (*Choice Function*) e FRR-MAB (*Fitness-Rate Rank Multi-Armed Bandit*) juntamente com suas maneiras de calcular a recompensa e função de escolha de heurísticas de baixo nível (LLHs).

Algoritmos Evolutivos são largamente utilizados para solucionar problemas na área de Engenharia de Software Baseada em Busca. Diversas são as aplicações de SBSE no problema de configuração de LPS com os mais variados propósitos como projeto, arquitetura, evolução, customização e teste. O foco principal deste trabalho é a derivação de produtos para o teste de LPS. Dessa forma, o próximo capítulo descreve trabalhos que utilizam alguns dos algoritmos descritos nesse capítulo para resolver problemas de Engenharia de Software, priorizando trabalhos que realizam o teste de LPS, e que estão mais relacionados ao presente trabalho.

# 4 TRABALHOS RELACIONADOS

A Engenharia de Software possui inerentemente características matemáticas e problemas para os quais é difícil de se alcançar uma solução, seja devido a presença de objetivos conflitantes, soluções desconhecidas ou mesmo a existência de um espaço de busca muito grande. Nas diferentes fases da Engenharia de Software, muitos problemas podem ser modelados como problemas de otimização e resolvidos por algoritmos de busca. Portanto, buscando respostas mais adequadas para essas situações, surgiu a linha de pesquisa denominada como Engenharia de Software Baseada em Busca (*Search Based Software Engineering* - SBSE) [38].

Neste contexto, nesse capítulo são apresentados alguns trabalhos encontrados na literatura de SBSE que se relacionam com os objetivos desse trabalho. Primeiramente é abordado o tema de teste de LPS baseado em busca, seguindo de otimização baseada em preferência em SBSE e por fim a utilização de hiper-heurísticas em SBSE.

## 4.1 TESTE DE LPS BASEADO EM BUSCA

Nesta seção são relacionados os principais trabalhos que utilizam algoritmos de busca para o teste de LPS a partir do FM.

O objetivo do trabalho de Henard et al. [40] é realizar a priorização de casos de teste de tal maneira que a forma em que estes são organizados possibilitem acelerar a descoberta de defeitos. Foi proposta uma abordagem baseada em similaridade para a geração e seleção de produtos para o teste de grandes LPSs através do critério *t-wise*. A ideia de similaridade, neste escopo, consiste em uma medida usada para comparar dois produtos. Os resultados obtidos pelo trabalho apontam que dois produtos dissimilares apresentam maiores chances de cobrir um maior número de t-conjuntos válidos (conjuntos criados pelo critério *t-wise*) do que dois produtos similares. O processo inicia com a utilização do Algoritmo Evolutivo (1+1) para gerar produtos válidos em que a similaridade é utilizada como valor de *fitness* para a seleção dos indivíduos mais aptos (que apresentam maior dissimilaridade entre si). São utilizados os algoritmos *Greedy Priorization* e *Near Optimal Priorization*. O processo é dado pela geração de produtos válidos e a priorização dos casos de teste, ou seja, a seleção dos conjuntos de produtos para o teste.

Em outro trabalho, Henard et al. [42] propõem a formulação de um problema multiobjetivo com função de agregação que visa a um conjunto final de produtos com a maior cobertura *pairwise* e o mínimo número de produtos e custo. Para tanto, os autores utilizam um algoritmo genético, propondo a representação para os indivíduos bem como operadores genéticos de cruzamento e mutação. Resolvedores SAT são utilizados para gerar apenas produtos válidos para compor o espaço de busca. Ou seja, a busca por um conjunto final ocorre em um espaço de busca que contém apenas produtos válidos para um diagrama em questão.

O teste baseado em mutação de LPS é abordado no mais recente trabalho de Henard et al. [41]. A abordagem proposta, inicia-se criando FM mutantes a partir do FM original válido. Então, um processo de busca baseado no Algoritmo Evolutivo (1+1) faz uso do FM original e

FM mutante para gerar um conjunto reduzido de dados de teste. A representação do indivíduo adotada consiste em um conjunto de configurações de teste e a função de *fitness*, consiste no escore de mutação de cada configuração do indivíduo.

No trabalho de Wang et al. [75] é abordada a minimização de casos de teste utilizando algoritmos genéticos baseados em uma função de agregação, levando em consideração os seguintes fatores: número de casos de testes, habilidade em revelar defeitos e cobertura *pairwise*. Foram avaliadas diferentes variações do AG considerando diferentes pesos na função de agregação. Foram utilizados três diferentes algoritmos que tratam esses pesos de diferentes maneiras: WBGA (pesos pré-definidos), WBGA-MO (possui um conjunto de pesos para seleção) e RWGA (pesos aleatórios normalizados). O principal objetivo é determinar um subconjunto reduzido de casos de teste, do conjunto total de casos de teste de uma LPS, para testar um determinado produto. Ao mesmo tempo busca-se atingir um alto nível de cobertura *pairwise* e um alto nível de detecção de defeitos.

Em um trabalho mais recente Wang et al.[87], apresentam uma abordagem de priorização de casos de teste no contexto de LPS. O problema é formulado visando à otimização de quatro objetivos: custo total para alocação de recursos de teste, número de casos de teste, cobertura *pairwise* e capacidade de detecção de defeitos dos casos de teste. Para resolver este problema, os autores utilizam três algoritmos, sendo eles, AVM (*Aternating Variable Method*) o qual representa os algoritmos de busca local, Algoritmo Genético e por fim o Algoritmo Evolutivo (1+1).

Também é proposta por Ensan et al.[26] uma abordagem para o problema de minimização de casos de teste para LPS. A ideia consiste em primeiro gerar aleatoriamente um conjunto de configurações das características, levando somente em consideração as combinações válidas. Essas configurações são utilizadas para formar a população inicial do algoritmo genético. A avaliação das soluções é realizada por uma função de *fitness* que leva em consideração duas métricas, sendo a cobertura de variabilidades que se refere ao total de pontos de variabilidade expressos em um determinado produto e, a segunda, a complexidade ciclomática, que se refere ao número de restrições de integridade que são aplicadas no processo de derivação de produtos de um determinado diagrama de característica. Mesmo com as configurações geradas sendo baseadas em configurações anteriores elas não são necessariamente válidas, dessa forma, é executado um processo para verificar as configurações geradas, selecionando somente as válidas para o processo. Esse processo é repetido até que a condição de parada seja satisfeita. O resultado final do processo, resulta em um conjunto de casos de testes e cada configuração consiste em um caso de teste que pode ser usado como uma aplicação, para o teste de LPS.

Lopez-Herrejon et al. [92] utilizam um algoritmo multiobjetivo para satisfazer o teste *pairwise*. O trabalho usa dois objetivos: minimizar o número de produtos de teste e maximizar a cobertura *pairwise*. Para tanto, os autores definem um valor fixo para o número de produtos de teste e uma relação com o número de características do FM. Os autores propõem um algoritmo mutiobjetivo que recebe um FM válido e como saída resulta em um conjunto de casos de teste expressos em uma fronteira de Pareto.

No trabalho de Matnei Filho e Vergilio [61] é proposta uma abordagem de otimização multiobjetivo para geração de casos de teste para o problema de teste de mutação de FMs. A abordagem inclui uma representação específica para o problema, assim como operadores genéticos. No trabalho, são apresentados experimentos utilizando dois (objetivos i e ii) e três (objetivos i, ii e iii) objetivos de minimização, sendo eles respectivamente: (i) número de casos de teste; (ii) número de mutantes mortos; e (iii) e número de pares de características cobertos (*pairwise*). A abordagem foi implementada fazendo-se uso de três algoritmos evolutivos mutiobjetivos, NSGA-II, SPEA2 e IBEA. A fase experimental foi conduzida utilizando quatro LPS diferentes: CAS, WeatherStation, Eshop e James. Os resultados obtidos foram comparados

com base nos indicadores de qualidade *Error Ratio*, *hypervolume*, e nos tempos de execução. Foi observado com os resultados, que os algoritmos NSGA-II, SPEA2 e IBEA obtêm um número reduzido de conjuntos de casos de teste quando comparados aos conjuntos totais de produtos válidos gerados. De maneira geral, todos os algoritmos obtiveram um bom desempenho considerando dois e três objetivos. Mas, o algoritmo NSGA-II se destacou no caso geral. Segundo os autores, uma vantagem da abordagem é oferecer ao testador um conjunto de boas soluções, com um número reduzido de produtos e altos valores de cobertura. Entretanto, o testador deve distinguir as melhores soluções de acordo com suas necessidades.

## 4.2 ALGORITMOS BASEADOS EM PREFERÊNCIA NA ÁREA DE SBSE

O trabalho de Ferreira et al. [33] apresenta resultados de um mapeamento da área de incorporação de preferências em SBSE. O objetivo do trabalho foi identificar a quantidade e o tipo de pesquisas que estão sendo feitas sobre as abordagens baseadas em preferência em SBSE e contribuir para o recente tema de pesquisa, que foi intitulado pelos autores de PSBSE (*Preference and Search-Based Software Engineering*). Ao todo, foram selecionados 40 trabalhos que abordam este tema, sendo divididos nas subáreas da engenharia de software: documentação e especificação (13 trabalhos); distribuição, manutenção e aperfeiçoamento (6 trabalhos); ferramentas e técnicas de projeto (14 trabalhos); e teste de software (7 trabalhos). Dentre os trabalhos citados, a abordagem mais utilizada consistiu em algoritmos evolutivos com um único objetivo. Na maioria dos estudos, é destacado que as preferências do usuário são fornecidas interativamente e, em muitos casos, as preferências do usuário são incorporadas na função de *fitness*. Os autores ainda destacam que foram encontradas poucas medidas de avaliação e trabalhos comparando abordagens existentes.

Dentre os trabalhos da área de PSBSE, são descritos a seguir os que abordam o teste de software e o contexto de LPS, que são os mais relacionados a esta dissertação.

Os trabalhos de Marculescu et al. [57, 58, 59] fazem parte do mesmo escopo, estando relacionados com a perspectiva da área de teste de software. O objetivo da pesquisa consiste em propor um sistema interativo que faz uma distinção entre os interesses da engenharia de software e do domínio de aplicação, permitindo que os especialistas de domínio interajam com o sistema a fim de selecionar os critérios de qualidade a serem utilizados. A ferramenta considera apenas a opinião do especialista de domínio, uma vez que cada domínio possui critérios específicos de qualidade e grande variação na modelagem do software.

Feldt [27] em seu trabalho, propõe um ambiente de desenvolvimento interativo onde os testes são criados em conjunto com o engenheiro de software que escreve o código do programa ou refina as especificações. O sistema usa as interações do engenheiro de software para ajudar a guiar a busca, porém com um efeito sobre a função de *fitness* ocorrendo de forma indireta. O objetivo da abordagem é expor características desconhecidas do software que está sendo desenvolvido, de tal modo, que seja possível o desenvolvedor verificar se correspondem às suas necessidades ou expectativas. Para isso, é empregada uma abordagem *biomimetic* de busca para encontrar testes que atendam a diferentes características. Segundo o autor, a abordagem proposta consegue determinar com êxito conjuntos de teste que são considerados poderosos e significativos.

No trabalho de Kalboussi et al. [47] é utilizado o método de ponto de referência para orientar a busca considerando múltiplos objetivos. Os autores fazem uso do PEMOA r-NSGA-II adaptando-o para o problema de agentes autônomos de muitos objetivos, denominando a

abordagem como *Preference-Based Many-Objective Evolutionary Testing Method* (P-MOET). A ideia principal é propor funções objetivo adicionais levando em consideração características específicas do software, visando a gerar casos de teste mais fortes capazes de revelar defeitos mais difíceis. Para este trabalho foram propostos sete objetivos. Os resultados obtidos demonstram que a abordagem proposta apresenta bons resultado, conseguindo ser melhor do que uma abordagem existente, baseada no algoritmo NSGA-II empregado neste problema. Além disso, o método é considerado um caso em especial devido a nova característica de incorporar a preferência do usuário, possibilitando assim, enfatizar alguns aspectos ou regiões de teste.

No trabalho de Mkaouer et al. [64] é proposta uma ferramenta para realizar recomendações interativas para refatoração de software, sendo realizadas adaptações dinamicamente e sugerindo refatorações ao desenvolvedor. Os autores fazem uso do algoritmo NSGA-II para determinar um conjunto de soluções que expressem as refatorações, visando à otimização de três objetivos, melhorar a qualidade do software, reduzir o número de refatorações e aumentar a coerência semântica. A inovação consiste em uma nova maneira de selecionar a(s) melhor(es) solução(ões) na fronteira de Pareto. Para tanto, utiliza-se uma maneira na qual ocorre uma interação direta com o desenvolvedor. O algoritmo de inovação começa encontrando os padrões/operações de refatoração que são mais frequentes entre o conjunto de soluções de refatoração não-dominadas. Dessa forma, as refatorações são classificadas e sugeridas ao desenvolvedor uma a uma, ficando a cargo do desenvolvedor, aprovar, modificar ou rejeitar cada refatoração sugerida. Este *feedback* é utilizado para atualizar o *rank* das refatorações sugeridas. Após uma certa quantidade de atualizações/modificações a serem realizadas no código, uma busca local é executada para atualizar e adaptar o conjunto de soluções de refatorações sugeridas pelo NSGA-II.

No âmbito de LPS, dois trabalhos foram encontrados, os quais empregam otimização interativa para configuração de produtos. No primeiro trabalho, Yamany et al. [91] propõem a ferramenta *Opti-Select* que adota uma nova técnica de exploração, que mescla a visão de especialistas com algoritmos de otimização, para produzir uma lista de soluções que melhor se adaptam às necessidades do usuário. O processo de otimização possui uma forma incremental. Depois de cada rodada, é permitido ao usuário selecionar as melhores soluções na fronteira de Pareto (melhores produtos) a fim de concentrar as buscas nas soluções desejadas. O algoritmo de otimização utilizado é o IBEA. O segundo trabalho [90], consiste em uma continuação do primeiro, sendo que uma nova ferramenta foi desenvolvida com base na anterior, denominada *Smart Opti-Select*. As principias diferenças com a versão anterior, estão no fato de a nova versão utilizar técnicas de aprendizado de máquina para entender as preferências do usuário, otimização multiobjetivo hibrida, mesclando resultados do algoritmo NSGA-II e IBEA e também a possibilidade do usuário estabelecer objetivos. Dessa forma, a nova versão da ferramenta se apresenta mais robusta e completa em relação à primeira.

## 4.3 HIPER-HEURÍSTICAS NA ÁREA DE SBSE

Ainda são poucos os trabalhos que usam hiper-heurística em SBSE. Alguns destes trabalhos são descritos nessa seção, com destaque para aqueles que abordam o teste de LPS.

No trabalho de Jia et al. [45] foi implementada uma hiper-heurística de aprendizado *on-line* para aprender e aplicar estratégias de teste combinatorial. O objetivo é obter um algoritmo genérico para aplicar este tipo de teste. Os resultados foram analisados e comparados com resultados das técnicas do estado da arte e com os melhores resultados obtidos na literatura. A hiper-heurística obteve um bom desempenho em problema restritos e irrestritos em vários casos.

Basgalupp et al. [3] propõem uma hiper-heurística *off-line* visando à geração de algoritmos que criam árvores de decisão. Estas árvores foram utilizadas na predição de esforço de software. Os resultados deste trabalho demonstram que os algoritmos gerados pela hiper-heurística foram melhores que os melhores resultados obtidos por alguns algoritmos do estado da arte.

Kumari et al. [52] utilizaram uma hiper-heurística para solucionar o problema de clusterização de módulos, e selecionar LLHs, compostas por operadores de seleção, cruzamento e mutação, durante o processo de otimização. Os resultados obtidos mostraram um melhor desempenho da hiper-heurística em relação a um algoritmo evolutivo convencional para todos os problemas investigados

No trabalho de Guizo et al. [35] é proposta uma hiper-heurística denominada HITO (*Hyper-heuristic for the integration and Test Order Problem*) para resolução do problema de estabelecer sequências de módulos para o teste de integração, utilizando o algoritmo NSGA-II. As LLH utilizadas, são compostas de um par de operadores de cruzamento e mutação. Os métodos de seleção aplicados foram CF e o *Sliding Multi-Armed Bandit*. Os resultados obtidos mostraram-se melhores que os obtidos pelo NSGA-II convencional.

Recentemente, o trabalho de Strickler et al. [81] apresenta uma abordagem utilizando hiper-heurística para derivar produtos para o teste de variabilidade em FMs. Para representar um indivíduo, é utilizada uma cadeia binária de produtos, onde o valor 0 significa que um produto não está presente no indivíduo e 1 o produto pertence ao indivíduo. Em relação às funções de *fitness*, são utilizados os mesmos objetivos descritos no trabalho de Matnei Filho e Vergilio [61], descrito anteriormente: cobertura *pairwise*, escore de mutação e conjunto reduzido de produtos.

Resumidamente o processo evolutivo da hiper-heurística, ocorre da seguinte maneira: (i) selecionam-se os pais; (ii) seleciona-se uma LLH de acordo com o método de seleção da hiper-heurística; (iii) aplica-se a LLH nos pais selecionados; (iv) avaliam-se as soluções; e (v) avalia-se a LLH aplicada.

Como heurísticas de baixo nível (LLHs) utilizadas pela hiper-heurística são adotados diferentes operadores de mutação e cruzamento. Ao todo, os autores fizeram uso de 12 heurísticas de baixo nível.

Para realizar a seleção das LLHs, os autores fizeram uso de dois métodos, sendo o FRR-MAB e uma maneira de seleção aleatória. Para avaliar a melhoria causada pela aplicação de uma determinada LLH, no método FRR-MAB, os autores utilizam uma atribuição de recompensa que pode assumir três valores, dependo da relação de dominância entre o pai $p$ e o filho $f$ [81]:

$$FIR_{i,t} = \begin{cases} 1 & \text{se } f \prec p, \\ 0 & \text{se } p \prec f, \\ 0.5 & \text{caso contrário} \end{cases} \tag{4.1}$$

Para comparar a abordagem proposta, Strickler et al. [81] utilizaram três algoritmos multiobjetivos tradicionais, sendo eles, NSGA-II, SPEA2 e IBEA. Segundo os autores a abordagem proposta utilizando hiper-heurística apresentou melhores resultados em relação as abordagens tradicionais.

Ainda considerando o mesmo problema e formulação da população e objetivos, no trabalho de Ferreira et al. [31] é proposta uma hiper-heurística aplicada com o algoritmo MOEA/D-DRA (*Multi-objective Evolutionary Algorithm Based on Decomposition with Dynamical Resource Allocation*) com três algoritmos baseados em UCB (*Upper Confidence Bound*). É destacado que os métodos de seleção propostos quando comparados com um método de seleção aleatório apresentam desempenho semelhantes, entretanto os métodos superam a versão canônica do MOEA/D-DRA.

Em outro trabalho, Ferreira et al. [32] propõem outra hiper-heurística que realiza a seleção dinâmica dos melhores operadores genéticos, considerando quatro objetivos para o problema: número de produtos, cobertura *pairwise*, escore de mutação e dissimilaridade de produtos. A abordagem é implementada e avaliada em quatro MOEAs: NSGA-II, SPEA2, IBEA e MOEAD/D-DRA, e dois métodos de seleção: aleatório e o UCB. Os resultados da avaliação mostraram que o algoritmo NSGA-II baseado em hiper-heurística gerou os melhores resultados, assim como o método de seleção FRR-MAB.

## 4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as principais abordagens encontradas na literatura referentes ao teste de LPS baseado em busca e também a utilização de preferência de usuário no âmbito de teste.

Devido ao problema de teste de LPS abordar diferentes fatores, a aplicação de abordagens multiobjetivo torna-se um opção mais promissora. Com base nos trabalhos descritos, observou-se uma predominância geral do algoritmo NSGA-II em relação aos demais EMOAs já utilizados no mesmo problema, ou seja, o EMOA NSGA-II foi considerado em um caso geral como a melhor opção para o problema abordado.

No mesmo contexto, o uso de hiper-heurísticas tem sido um ponto de destaque na área de SBSE. Abordagens HH fornecem uma maior flexibilidade e autonomia para o testador configurar e implementar abordagens multiobjetivo de uma maneira genérica. Contudo, é enfatizada a grande eficiência da utilização de HH para o teste de LPS. Pode-se destacar que o uso do algoritmo NSGA-II com as abordagens HH (NSGA-II-HH), apresentou na maior parte dos casos, os melhores resultados com relação ao EMOAs e outras abordagens HH.

Um fator limitante encontrado no uso de abordagens que aplicam EMOAs e também HH ocorre devido ao grande número de soluções desnecessárias do ponto de vista do testador, resultantes do processo de busca. O processo de escolha da melhor soluções pode ser uma tarefa complexa e massante, consumindo um grande tempo e esforço para o testador. Com base nisso, a utilização de abordagens que incorporam as preferências do usuário no processo de busca torna-se um fator de relevância.

Poucos trabalhos levam em consideração as preferências do usuário durante o teste de software. Mais especificamente, não foram encontrados trabalhos que consideram PEMOAs no teste de LPS, e do mesmo modo, hiper-heurísticas baseadas em preferência. Tendo em vista a grande contribuição do uso de tais abordagens para facilitar a tarefa do testador no teste de LPS, surge um espaço para pesquisa nessa área.

Contudo, levando em consideração os bons resultados obtidos pelo EMOA NSGA-II para o teste de LPS, foi adotado para este trabalho PEMOAs que considerem ou sejam baseados neste algoritmo. Tendo em vista que tal fator, pode possibilitar que os bons resultados já obtidos, sejam ainda melhorados quando levadas em consideração as preferências do usuário no processo de busca no problema descrito. Tendo em vista a grande variedade de maneiras de incorporar a preferência do testador, leva-se em consideração o método de ponto de referência, tendo em vista a facilidade para o testador expressar suas preferência por esse método. Pois, por muitas vezes, são pré estabelecidas na fase de teste, os valores desejados para a cobertura de critérios, o custo relativo do teste, entre outros fatores. Dessa forma, o testador tem suas preferências incorporadas de forma natural, evitando ainda o problema de fadiga, tendo em vista que as informações são repassadas a priori, e obtendo apenas o conjunto de soluções que melhor satisfaçam suas necessidades para o teste. Tendo em vista as condições aqui expostas, os próximos capítulos detalham a abordagem proposta e o estudo experimental para avaliação da mesma.

# 5 ABORDAGEM PROPOSTA

Neste capítulo é definida a abordagem proposta com o objetivo de incorporar as preferências do usuário em abordagens multiobjetivo que derivam produtos para o teste de LPS.

Primeiramente é dada uma introdução geral, e são descritos os seus principais aspectos: representação da população, funções objetivo e aspectos de implementação dos PEMOAs e HH considerando três métodos de seleção de LLHs: CF, FRR-MAB e método aleatório.

## 5.1 DEFINIÇÃO DA ABORDAGEM

A abordagem tem como objetivo geral a criação de um conjunto de soluções que melhor expresse o interesse do testador, e ao mesmo tempo satisfaça os critérios de teste utilizados no teste de variabilidades de LPS.

O teste de variabilidade é responsável por garantir que os produtos derivados do modelo de características correspondam aos requisitos esperados. Assim, a abordagem é baseada no Modelo de Características (FM), que é amplamente adotado na indústria.

A Figura 5.1 mostra um exemplo de FM para uma LPS do domínio de telefonia móvel [5] que será usado nesta seção para ilustrar a abordagem. Idealmente, todos os produtos possíveis derivados do FM devem ser testados. O uso de um critério de teste pode ajudar a seleção dos produtos mais representativos.

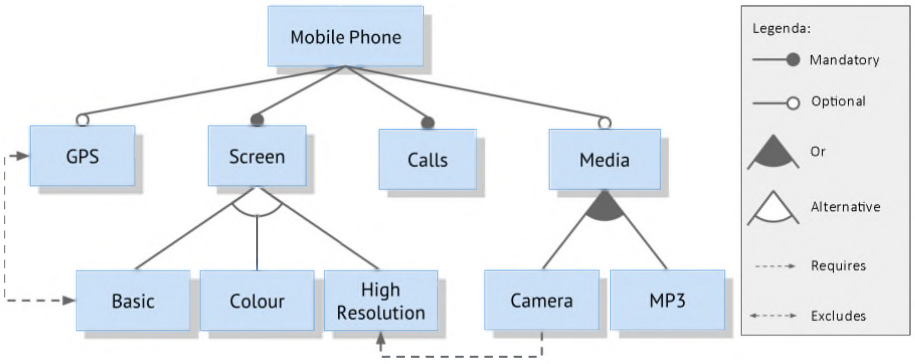



Figura 5.1: FM para uma LPS de telefonia móvel (adaptada de [32]).

Um critério de teste também pode fornecer uma maneira de selecionar e avaliar os dados do teste, que neste contexto são os produtos a serem testados. Então, o problema é determinar um conjunto mínimo de produtos que satisfaçam determinados critérios de teste. Esse conjunto deve ter uma grande capacidade de revelar defeitos, que podem estar presentes no FM, com um custo reduzido (ou ótimo).

Ao analisar as características do nosso problema, observa-se em muitos casos que existe um plano de teste que contém a cobertura desejada para os critérios de teste, bem como um

limite para o número de produtos a serem testados, devido a restrições de custo. Desta forma, mesmo sem qualquer conhecimento sobre o espaço de soluções, não é difícil para o testador especificar valores desejados para os objetivos a serem avaliados. Considerando este fato, a abordagem utiliza dois PEMOAs baseados no método do ponto de referência, sendo eles o algoritmo R-NSGA-II [21] e o algoritmo r-NSGA-II [77].

Nesse mesmo contexto, e tendo em vista os bons resultados que a utilização de hiper-heurísticas tem proporcionado durante a solução deste problema, a abordagem inclui a adaptação de uma HH já estabelecida na literatura [81], para permitir sua aplicação com os PEMOAs utilizados.

### 5.1.1 Representação da População

Um indivíduo da população é definido como um conjunto de dados de teste (produtos) derivados para uma determinada LPS em teste. A representação do indivíduo consiste em um vetor *booleano* $T = \{P_1, P_2, P_3, ..., P_n\}$, onde $n$ representa o total de produtos derivados pela LPS. Dessa forma, o valor 1 para o índice $i$ do vetor representa que o produto $i$ está selecionado, e 0 representa que o produto $i$ não está selecionado. A Figura 5.2 apresenta uma síntese da representação de um possível indivíduo. Os indivíduos $P_2$ e $P_4$ não estão selecionados, já $P_1$, $P_3$ e $P_5$ são selecionados, a mesma convenção é utilizada para representar as características presentes em cada produto.

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | ... | $P_n$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | ... | ... |

Figura 5.2: Representação da população.

### 5.1.2 Funções Objetivo

Foram definidos três objetivos para o cálculo de *fitness* no indivíduo ($\Delta$). Esses objetivos foram baseados nos trabalhos relacionados [61, 81]. Todos os objetivos foram modelados como um problema de minimização.

O primeiro objetivo corresponde ao total de produtos presentes em um indivíduo ($S_\Delta$). Esse número é expresso entre a razão do número total de produtos presentes no indivíduo ($n_\Delta$) e o número total de produtos válidos gerados pelo diagrama de característica ($nt$). Para calcular o número de produtos válidos, foi utilizado o framework *Feature Model Analyzer* (FaMa) [85]. Tal objetivo visa a determinar o menor número de produtos possível para um indivíduo respeitando ao mesmo tempo os demais objetivos. Esse calculo é apresentado na Equação 5.1.

$$S_\Delta = \frac{n_\Delta}{nt} \tag{5.1}$$

O segundo objetivo corresponde à cobertura dos diagramas mutantes gerados. Um diagrama mutante é dito morto (coberto) por um determinado caso de teste (produto) quando o mesmo é considerado válido para o diagrama original e inválido para o diagrama mutante, ou válido para o diagrama mutante e inválido para o diagrama original. Para isso, foi utilizado o conjunto de operadores de mutação e a ferramenta FMTS proposta por Ferreira et al. [29]. Esta ferramenta funciona com o FaMa [85], que é responsável pela validação dos modelos.

Dessa forma, objetiva-se minimizar a quantidade de mutantes que não são mortos (vivos) no processo de teste. Como o escore de mutação corresponde ao percentual de cobertura, o valor a ser minimizado corresponde a um menos o escore de mutação.

Para efetuar o cálculo do escore de mutação é levando em consideração o número de mutantes vivos ($A_\Delta$), dado com relação entre o número de mutantes mortos ($DM_\Delta$) e o número total de mutantes gerados $AM$. A Equação 5.2 apresenta o cálculo do escore de mutação.

$$A_\Delta = 1 - \frac{DM_\Delta}{AM} \tag{5.2}$$

A Figura 5.3 apresenta um exemplo de mutante para o FM da Figura 5.1. O operador de mutação altera uma relação do tipo "*requires*"para uma relação do tipo "*excludes*", ocorrendo entre as características HD e Câmera. Desta forma, um produto que contém ambas as características é válido para o FM original e inválido para o mutante correspondente, sendo este último considerado morto.

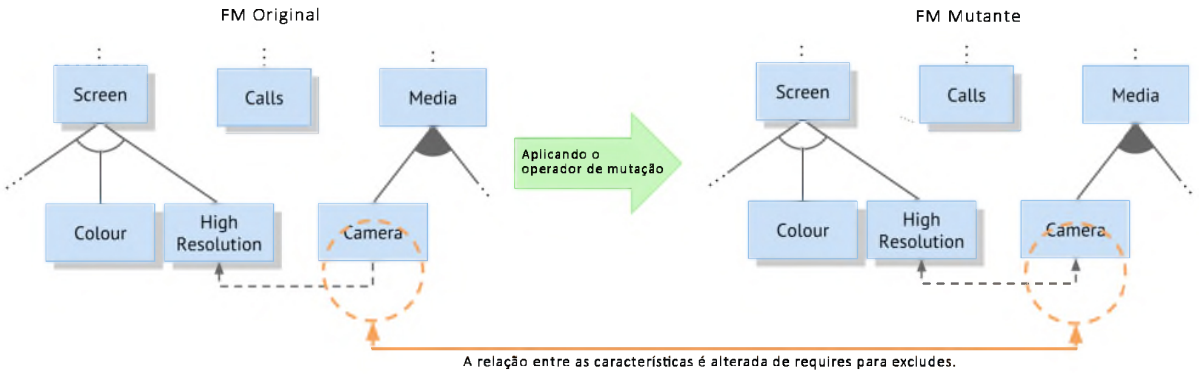



Figura 5.3: Exemplo de um mutante criado a partir da LPS telefone celular (adaptada de [32]).

Nesse sentido, o produto da Figura 5.4 mata o mutante, uma vez que é válido para o FM original e é inválido para o mutante.

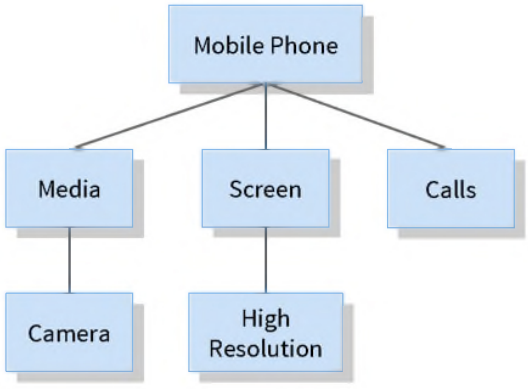



Figura 5.4: Exemplo de um produto válido (adaptada de [32]).

Por fim, o terceiro objetivo corresponde à cobertura de pares gerados para satisfazer o critério de teste *pairwise*. Um par é dito coberto quando determinado produto possui as duas características presentes em um par. Nesse objetivo procura-se minimizar a quantidade de pares não cobertos. Dessa forma assim como no objetivo anterior, reduz de um o valor encontrado para os pares em questão.

A cobertura *pairwise* ($P_\Delta$) corresponde à relação entre o número de pares cobertos pelo indivíduo ($PC_\Delta$) e o número total de pares válidos ($P$) [61]. A Equação 5.3 apresenta o cálculo da cobertura.

$$A_\Delta = 1 - \frac{PC_\Delta}{P} \tag{5.3}$$

Para exemplificar, ao analisar a Figura 5.1, o par de características (GPS, Básico) é inválido e não deve ser requerido. Se levar em consideração apenas as variabilidades, observa-se que o produto da Figura 5.4 inclui o par (Alta Resolução, Câmera). Neste trabalho, foi utilizada a ferramenta *Combinatorial tool* para derivar os pares. Esta ferramenta implementa o algoritmo *Automatic Efficient Test Generator* (AETG) [17].

## 5.2 ASPECTOS DE IMPLEMENTAÇÃO

Nesta seção são apresentados os aspectos de implementação da abordagem proposta. A implementação da abordagem ocorreu em duas etapas. A primeira etapa, descreve os PEMOAs implementados, assim como os dados gerais para a execução de cada algoritmo. A segunda parte, consiste em descrever os aspectos referentes a integração da hiper-heurística para considerar as preferências do usuário. Tais informações, são apresentadas nas subseções seguintes.

Para realizar a implementação das abordagens foi utilizada a linguagem de programação Java e o ambiente de desenvolvimento Eclipse IDE. Além disso, ainda foi utilizado o *framework* jMetal [24], em sua versão mais recente. O jMetal é um *framework* para manipulação de problemas de otimização. É fornecida uma estrutura com formulações de soluções, problema, indicadores de qualidade e algoritmos.

Como entrada do problema, é utilizada uma matriz composta pelo conjunto de mutantes gerados durante o teste de mutação e o conjunto de dados de teste (produtos) que matam os mutantes. Essa matriz possui uma representação binária, onde cada coluna representa um determinado caso de teste e cada linha um mutante gerado. Se o elemento com indicie $M_{i,j}$ tiver o valor 0, representa que o determinado dado de teste não mata o produto, já se o valor for 1, representa que o dado de teste mata o determinado mutante. Como mencionado anteriormente essa matriz é obtida com a ferramenta FMTS [30].

A Tabela 5.1 apresenta a estrutura da matriz de entrada para o problema. Na tabela as linhas representam os mutantes e as colunas representam os produtos (dados de teste). É importante destacar que os mutantes informados na matriz de entrada, são mortos por pelo menos um dado de teste. Dessa forma, os mutantes que são identificados como equivalentes, pela ferramenta FMTS, não são considerados. Os mutantes equivalentes derivam os mesmos produtos que o FM em teste.

Tabela 5.1: Matriz de entrada para o problema.

| Mutantes | Produto1 | Produto2 | Produto3 | Produto4 | Produto5 |
|---|---|---|---|---|---|
| Mutante1 | 0 | 1 | 1 | 1 | 0 |
| Mutante2 | 1 | 1 | 1 | 0 | 0 |
| Mutante3 | 1 | 1 | 0 | 1 | 0 |
| Mutante4 | 0 | 0 | 1 | 1 | 0 |
| Mutante4 | 1 | 1 | 0 | 0 | 1 |

Além da matriz de entrada, são necessários os dados para realização do cálculo de cobertura do teste *pairwise*. Como mencionando anteriormente, para isto foi utilizada a ferramenta

*Combinatorial tool* que implementa o algoritmo AETG para geração dos pares. A utilização de tal ferramenta proporciona como saída uma matriz onde cada linha representa um par e cada coluna representa um dado de teste (produto). Da mesma maneira que a matriz de mutantes, se a posição $M_{i,j}$ contiver o valor 0, o par não é coberto pelo determinado dado de teste, senão, se o valor for 1, o par é coberto. Somente pares válidos são considerados.

### 5.2.1 Implementação dos PEMOAs

Nesta subseção apresentam-se os aspectos gerais da implementação dos algoritmos baseados em preferência, mais especificamente R-NSGA-II e r-NSGA-II. Para ambos os algoritmos, foi utilizada a mesma representação da população, mesmas funções de *fitness* operador de seleção e operadores genéticos. O operador de seleção e operadores genéticos utilizados são descritos a seguir. Tais operadores foram selecionados com base no trabalho de Strickler et al. [81] devido aos bons resultados obtidos quando aplicados no mesmo problema abordado.

Para realizar a seleção dos melhores indivíduos para sofrerem as operações genéticas, utiliza-se o operador de torneio binário. Nesse tipo de seleção os dois melhores indivíduos de uma população são selecionados para a próxima geração [18]. Caso ocorra o cruzamento, os dois indivíduos são recombinados entre si resultando na geração de dois novos filhos. E caso ocorra a mutação, os dois indivíduos selecionados sofrem alterações genéticas.

Os operadores genéticos são utilizados para inserir diversidade na população. O operador de mutação a ser utilizado, consiste na mutação por troca de bit (*Bit Flip*). Basicamente, é selecionada uma posição aleatória nas colunas do indivíduo, se esta posição contiver o valor 0, este é trocado para o valor 1, e assim vice-versa. O operador de cruzamento utilizado consiste no cruzamento uniforme. Neste operador, cada gene do descendente é criado copiando o gene correspondente de um dos pais, escolhidos de acordo com uma máscara de cruzamento gerada aleatoriamente. Onde houver 1 na máscara de cruzamento, o gene correspondente será copiado do primeiro pai e, onde houver 0 será copiado do segundo. Para gerar o segundo descendente o processo é repetido com os pais trocados.

De modo geral, a implementação de ambos os algoritmos, R-NSGA-II e r-NSGA-II, seguiu a estrutura de funcionamento original especificada respectivamente para cada algoritmo nas Subseções 3.3.1 e 3.3.2.

Além das duas matrizes de entrada, é necessário que o DM informe alguns parâmetros para o funcionamento dos algoritmos, tais como: tamanho da população, número máximo de gerações, taxa de cruzamento, taxa de mutação, número máximo de gerações, número de produtos que serão gerados (quando necessário), o ponto de referência, o vetor de pesos para dar uma maior enfase na região de interesse do DM e também os parâmetros que limitam a diversidade de soluções próximas ao ponto de referência.

O ponto de referência, no caso do algoritmo r-NSGA-II, consiste em um vetor onde cada coluna representa o ponto de preferência para um objetivo, dessa forma, a junção de todos os valores para os objetivos descritos forma uma localização dentro do espaço de objetivos. Já no caso do R-NSGA-II, devido à possibilidade de trabalhar com mais de um ponto de referência ao mesmo tempo, é utilizada uma matriz, onde cada linha corresponde a um determinado ponto de referência no espaço objetivo.

Com todos os dados descritos anteriormente, o processo de execução do algoritmo pode ser iniciado. Para ambos os algoritmos baseados em preferência, R-NSGA-II e r-NSGA-II, o processo geral de otimização segue a mesma estrutura. Primeiramente os indivíduos da população inicial são gerados com base na seleção de um número aleatório de produtos do conjunto em questão. A partir de então, o processo de evolução se inicia, respeitando as características de cada

algoritmo. De forma geral, irão ocorrer $n$ gerações baseadas no número máximo de avaliações, podendo o processo ser reaplicado caso as preferências do usuário não sejam satisfeitas. Durante o processo evolutivo são aplicados os operadores genéticos, com a finalidade de inserir novo material genético na população. Para cada geração um valor de *fitness* é atribuído ao indivíduo, esse valor é calculado com base em ambas as matrizes informadas anteriormente e as funções descritas na Subseção 5.1.2. Ao final da execução obtêm-se conjuntos de produtos que melhor expressem as preferências do DM.

### 5.2.2 Integração da Hiper-heurística

Esta seção apresenta os aspectos relevantes para integração da HH com as preferências do usuário. De maneira geral algumas alterações especificas foram realizadas em cada método de seleção HH para que fosse possível trabalhar com os PEMOAs.

A integração do componente HH, atribuição de crédito e método de seleção em PEMOAs é um procedimento fácil. A HH deve gerenciar a aplicação dos operadores de busca durante o processo evolutivo, com a finalidade de obter soluções em uma região de interesse (ROI) estipulada pelo testador. Portanto, antes da aplicação de um operador, o método de seleção é chamado. Em seguida, o operador é aplicado e, no final, a atribuição de crédito é executada.

Para tal finalidade, adota-se como base a hiper-heurística proposta por Strickler et al. [81], descrita anteriormente na Subseção 4.3. Tal HH faz uso dos métodos de seleção FRR-MAB e do método aleatório. Além desses já implementados, foi também implementado o método de seleção Choice Function (CF).

Ao todo, foram consideradas 12 heurísticas de baixo nível, originadas pela combinação de operadores genéticos. Essas LLHs, já haviam sido utilizadas na versão original da hiper-heurística de Strickler et al. [81]. A Tabela 5.2 apresenta as LLHs utilizadas.

Tabela 5.2: Composição das heurísticas de baixo nível (LLHs).

| Mutação | Cruzamento | | |
|---|---|---|---|
| | Um Ponto | Dois Pontos | Uniforme |
| - | h1 | h2 | h3 |
| *Bit Flip* | h4 | h5 | h6 |
| *One Change* | h7 | h8 | h9 |
| *Swap* | h10 | h11 | h12 |

Foram utilizados três operadores de cruzamento e três operadores de mutação: (i) cruzamento de um ponto, cruzamento de dois pontos e cruzamento uniforme; e (ii) *Bit Flip*, *One Change* e *Swap*. Cada célula do meio da tabela representa uma LLH, formada pela combinação de um operador de cruzamento (coluna) e um operador de mutação (linha). Além disso, ainda pode ocorrer de uma LLH não conter um operador de mutação, como é o caso das heurísticas h1, h2 e h3.

Entretanto, algumas alterações específicas foram necessárias em cada algoritmo para possibilitar a integração com o PEMOA. Mais especificamente, as alterações ocorreram somente para a hiper-heurística utilizando o algoritmo r-NSGA-II, que utiliza o conceito de r-dominância.

Tal alteração no algoritmo HH tradicional é necessária devido às características do algoritmo r-NSGA-II, tendo em vista que a utilização da dominância de Pareto para atribuição de credito dos operadores pode acabar privilegiando as soluções que não estão próximas ao ponto de referência informado pelo testador, o que para essa abordagem, não se torna interessante.

A inserção do conceito de r-dominância ocorreu nos dois métodos de seleção utilizados. Tanto para o algoritmo FRR-MAB, quanto para o CF, a atribuição da recompensa ou escore de aplicação de cada LLH, baseou-se na dominância de Pareto, seguindo ainda como base o trabalho de Guizzo et al. [35]. De maneira geral, a dominância de Pareto foi substituída pela relação de r-dominância. As subseções seguintes apresentam como esse processo foi realizado em cada método de seleção.

Quando a integração do método é realizada com o algoritmo R-NSGA-II, é utilizada para a atribuição de recompensa ou o valor de escore, o conceito de relação tradicional de dominância de Pareto.

### Integração *Fitness-Rate-Rank based Multi-Armed Bandit* (FRR-MAB)

Como descrito anteriormente, o FRR-MAB é composto por duas etapas, sendo elas a escolha de LLHs e o método de atribuição de crédito. A escolha da LLH é baseada nos valores de FRR (*Fitness Rate Ranking*). Os valores de FRR são gerados pelo método de atribuição de crédito empregado pelo FRR-MAB. Estes valores têm como função avaliar a qualidade de uma LLH de forma normalizada. Além disso, os valores de FRR são calculados de forma que a influência da LLH que tenha os melhores resultados possa ser incrementada.

A atribuição de crédito é uma parte necessária para o funcionamento do FRR-MAB. No trabalho de Guizzo et al. [35] é utilizado o conceito de dominância de Pareto para tal tarefa, a melhoria causada pela aplicação de uma LLH é avaliada após o relacionamento de dominância. Com base nisso, propõe-se neste trabalho utilizar uma alteração neste método para considerar o conceito de r-dominância para à atribuição de crédito.

$$FIR_{i,t} = \frac{1}{|P|.|C|} * \sum_{p \in P} \sum_{c \in C} \begin{cases} 1 & \text{se } c \text{ r-domina } p, \\ 0 & \text{se } p \text{ r-domina } c, \\ 0.5 & \text{outro caso} \end{cases} \tag{5.4}$$

Para este algoritmo a atribuição de crédito usa uma janela deslizante (*Sliding Window* - SW) com tamanho fixo. A ideia é armazenar o valor de FIR da LLH usada mais recentemente na cauda da SW, enquanto a gravação mais antiga (o item na cabeça da fila) é removida para manter o tamanho da janela constante. Assume-se que a SW garante que os valores de FIR armazenados refletem a situação atual da pesquisa.

### Integração Choice Function (CF)

Para realizar o cálculo do escore de uma LLH, foi utilizada a função expressa pela Equação 3.9. De maneira geral, o processo evolutivo adotado pelo CF segue uma sequência de dois passos, que descrevem a forma como é calculado o score de cada LLH até o conjunto final de soluções.

Considerando o primeiro passo, cada heurística de baixo nível possui um valor de performance instantânea calculado por $f_1$ no intervalo [0, 1] e um valor de tempo calculado por $f_2$ no intervalo [0..+$\infty$], que definem respectivamente a última performance da heurística de baixo nível e quantos cruzamentos passaram desde a sua última execução. Quanto maior o valor de $f_1$, melhor a heurística de baixo nível se saiu. No segundo passo, ambos valores são inicializados com 0 e posteriormente cada heurística é aplicada uma vez para atribuir uma performance inicial. Esses valores são atualizados toda vez que uma heurística de baixo nível é aplicada. Basicamente, a cada atualização, o valor $f_2$ da heurística aplicada é zerado e o valor f3 das demais é incrementado em 1. Já para $f_1$, a sua atualização é feita utilizando apenas os pais e os filhos envolvidos no cruzamento e na mutação, de acordo com a Equação 5.4.

### Método aleatório

O método aleatório realiza a seleção de LLHs sem levar em consideração nenhuma informação referente às aplicações anteriores da LLH. Dessa forma, para realizar a seleção por este método, é sorteado um valor aleatório que representa um índice do vetor onde estão armazenadas cada LLH. Então, a LLH armazenada no respectivo índice, é aplicada no momento atual da busca.

### Aspectos Gerais

O Algoritmo 4 apresenta com foi realizada a integração do PEMOA r-NSGA-II com a HH, estabelecendo um novo algoritmo denominado aqui r-NSGA-II-HH. Inicialmente, uma população $P_0$ é gerada de maneira aleatória, com um tamanho $N$. Em seguida, é aplicada a seleção por torneio, definindo os pais que sofreram as operações genéticas. Se a HH aplicada for a CF, são atualizados o tempo de espera e o ranking das LLHs. Se o método for FRR-MAB, a qualidade e recompensas das LLHs são atualizadas no estado atual da janela de tempo. A aplicação da LLH proporciona a criação da primeira população de filhos $Q_0$.

A cada geração do algoritmo, os indivíduos das populações pai $p_i$ e filho $Q_i$ obtidos são ordenados e classificados de acordo com o conceito de r-dominância entre as soluções, formando diversas fronteiras (linhas 20 a 21). A primeira fronteira corresponde às soluções r-não-dominadas de toda a população, a segunda é constituída por todas as soluções que passam a ser r-não-dominadas, após a retirada de soluções da primeira fronteira, a terceira fronteira é composta por soluções que passam a ser r-não-dominadas após a retirada das primeira e segunda fronteiras, e assim consecutivamente até que todas as soluções estejam classificadas em alguma fronteira.

Para formar a população $P_{i+1}$, as soluções mais espalhadas são escolhidas usando o procedimento Calcula_CrowdingDistance pelo ponto $F_j$. Este procedimento retorna o valor de *crowding distance* para cada solução que indica o grau de aglomeração da solução.

Após a seleção de $N$ pontos de $P_{i+1}$, esta população é classificada com base na relação de r-dominância e *crowding distance* (operador $\geq_s$). Então, apenas as melhores soluções $N$ de $P_{i+1}$ permanecem na população. O último passo no ciclo evolutivo é a geração de uma nova população $Q_{i+1}$ aplicando a seleção por torneio e as LLHs escolhidas. Novamente, se a HH é CF, são atualizados o tempo de espera e o ranking, e se for FRR-MAB, a qualidade e recompensas das LLHs são atualizadas no estado atual da SW.

Também foi implementada a versão do algoritmo R-NSGA-II-HH que segue a mesma estrutura do Algoritmo 4. A única diferença está na linha 21, na qual ao invés de se aplicar o conceito de r-dominância, é aplicado do conceito de dominância de Pareto tradicional.

## 5.3 CONSIDERAÇÕES FINAIS

Na literatura, EMOAs foram utilizados para derivar produtos para o testes de LPS. Sendo que os melhores resultados foram obtidos pela combinação desses algoritmos com HHs. Para permitir a incorporação das preferências do usuário em tais algoritmos, este capítulo introduziu uma abordagem que possui as seguintes características.

A abordagem incorpora as preferências do testador a priori utilizando algoritmos baseados em ponto de referência (algoritmos R-NSGA-II e r-NSGA-II). Ambos os algoritmos são baseados no algoritmo NSGA-II. Portanto, a abordagem é multiobjetivo e permite a otimização

**Algoritmo 4:** Pseudocódigo do r-NSGA-II-HH.

```
1  início
2      P_0 = Q_0=0 Inicializa as populações P_0 e Q_0
3      i = 0 Inicializa o número de gerações
4      Gerar aleatoriamente população Q_0
5      para i = 0 até max_avaliações faça
6          Seleção por torneio binário
7          se CF então
8              tempo_de_espera = calcula_tempo_de_espera()
9              ranking = CalculaRanking()
10             op = ChoiceFunction();
11         fim
12         senão se FRRMAB então
13             Credit_Assignment(SW)
14             op = LLH_Seleção(SW)
15             Calcular_recompensa(op, SW)
16         fim
17         senão
18             op = LLH_Seleção() Seleciona uma LLH aleatoriamente
19         fim
20         U_i = P_i ∪ Q_i
21         Aplica seleção de r-dominância em U_i
22         P_{i+1} = 0
23         while |P_{i+1}| ≤ N do
24             Calcule_CrowdingDistances Para F_j
25             P_{i+1} = P_{i+1} ∪ F_j
26         end
27         Ordena P_{i+1} usando o operador ≥_s
28         P_{i+1} = P_i + 1[0 : (N-1)] Escolher N primeiros elem. de P_{i+1}
29         Gerar a população Q_{i+1}
30     fim
31 fim
```

de diferentes fatores que impactam o problema. Para o contexto deste trabalho, são utilizados três fatores, sendo a redução do número de produtos, o escore de mutação e a cobertura *pairwise*.

Estas escolhas se justificam pois o algoritmo NSGA-II apresentou os melhores resultados da literatura. Além disso, o método de ponto de referência é adequado para o problemas pois não é difícil para o testador fornecer um ponto, visto que existe na maioria das vezes um limite de custo e um nível de cobertura para os critérios, estabelecidos pelo plano de teste. A abordagem a priori apresenta a vantagem de não requerer várias interações do usuário como na abordagem interativa, o que pode causar fadiga.

Para permitir o uso desses algoritmos com a HH, também foi descrita uma proposta de integração destes algoritmos com três métodos de seleção, sendo eles, CF, FRR-MAB e um método de seleção aleatória de LLHs. Foi proposto o uso do conceito de r-dominância para seleção das LLHs em conjunto com o algoritmo r-NSGA-II.

A abordagem tem o objetivo de incorporar as preferências do usuário na busca por soluções concentradas na ROI e diminuir o número de soluções que não são interessantes do

ponto de vista do testador. Estes objetivos foram avaliados em um estudo experimental descrito no próximo capítulo.

# 6 AVALIAÇÃO EXPERIMENTAL

Neste capítulo são descritos os resultados da avaliação da abordagem proposta e dos algoritmos implementados. Primeiramente, é descrito como foram organizados os experimentos, incluindo questões de pesquisa, indicadores de qualidade, LPSs utilizadas, e com foram definidos os RPs e os parâmetros dos algoritmos. Ao final, os resultados são apresentados com o objetivo de responder às questões de pesquisa propostas.

## 6.1 QUESTÕES DE PESQUISA

A hipótese deste trabalho é que a abordagem proposta é capaz de gerar automaticamente um melhor conjunto de soluções levando em consideração as preferências do DM, dadas através do método de ponto de referência, evitando assim, soluções não interessantes. Além disso, deseja-se comparar o desempenho dos PEMOAs descritos anteriormente, considerando diferentes ROIs e objetivos. De acordo com a hipótese descrita, o experimento foi orientado pelas seguintes questões de pesquisa (QP):

**QP-1:** Como os resultados dos PEMOAs se comparam aos resultados obtidos pelo algoritmo tradicional NSGA-II?

**QP-2:** Qual é o melhor método de seleção para os algoritmos R-NSGA-II-HH e r-NSGA-II-HH?

**QP-3:** Como se compararam os resultados obtidos pelos algoritmos R-NSGA-II-HH e r-NSGA-II-HH, utilizando o melhor método de seleção identificado na QP-2, com relação aos PEMOAs?

**QP-4:** Como se comparam os resultados obtidos pelos algoritmos R-NSGA-II-HH e r-NSGA-II-HH, utilizando o melhor método de seleção identificado na QP-2, em relação ao algoritmo NSGA-II-HH com o método FRR-MAB, identificado como sendo o melhor da literatura [32, 81]

Para responder às questões de pesquisa e avaliar o uso da abordagem com diferentes conjuntos de objetivos, dois experimentos foram criados. O primeiro experimento foi realizado considerando uma formulação com 2 objetivos (F2O) relacionados ao número de produtos e o escore de mutação. O segundo experimento foi realizado considerando uma formulação com 3 objetivos (F3O) relacionados ao número de produtos, ao escore de mutação e cobertura *pairwise*. O objetivo deste último experimento é ter um espaço de busca maior para aumentar a complexidade da busca, bem como a dificuldade em encontrar boas soluções.

Nas seções a seguir, são descritos os indicadores de qualidade, a organização dos experimentos, FMs, RPs e parâmetros usados.

### 6.1.1 Indicadores de Qualidade

A análise de desempenho dos algoritmos é realizada através de indicadores de qualidade. Muitos indicadores requerem uma Fronteira de Pareto (*Pareto Front*), porém, para problemas do mundo real essa fronteira é desconhecida ou raramente existe. Para tanto, a estratégia mais comum utilizada [31, 81, 95] é a construção de uma Fronteira de Pareto Ótima ($PF_{true}$), composta da união de todas as soluções obtidas por todos os $k$ algoritmos em todas as suas $n$ execuções realizadas.

Por esse motivo foram definidos três conjuntos de soluções para a análise de desempenho dos algoritmos:

- $PF_{approx}$: este conjunto corresponde a uma aproximação do conjunto de Pareto Ótimo, que representa o conjunto de possíveis soluções não dominadas encontradas para determinado problema. Em cada execução de um PEMOA, um conjunto $PF_{approx}$ é obtido;

- $PF_{known}$: conjunto composto pelas melhores soluções encontradas por um determinado PEMOA para um problema. Em geral um PEMOA é executado várias vezes para que seu comportamento seja observado, dessa forma, o conjunto $PF_{known}$ é gerado pela união de todos os conjuntos $PF_{approx}$ obtidos pelo PEMOA, descartando as soluções dominadas e repetidas;

- $PF_{true}$: representa a Fronteira de Pareto Ótima para o problema. No nosso caso, esse conjunto é desconhecido. Devido a isso, e seguindo a literatura [31, 81, 95], este conjunto foi formado por todos os conjuntos $PF_{known}$ obtidos de diferentes algoritmos removendo as soluções dominadas e repetidas. O conjunto $PF_{true}$ é, de fato, uma aproximação da fronteira real.

Neste contexto, foram utilizados alguns indicadores de qualidade considerados relevantes para o escopo deste trabalho, sendo eles: i) Hypervolume em conjunto com R-Metric (R-HV); ii) Distância Euclidiana (DE) ; e iii) número médio de soluções e porcentagem de soluções na ROI. Esses indicadores são descritos nas subseções seguintes.

#### Hipervolume com R-Metric (R-HV)

No âmbito deste trabalho, utilizou-se o indicador de qualidade Hipervolume em conjunto com a R-Metric (R-HV). Esta é baseada no trabalho desenvolvido por Li et al. [46] e consiste em uma maneira sistemática de adaptar indicadores de qualidade para avaliar quantitativamente o desempenho de um PEMOA usando o método de ponto de referência. A ideia geral desta métrica é pré-processar as soluções preferidas de acordo com uma abordagem de tomada de decisão multi-critério antes de usar uma métrica regular para avaliar o desempenho das soluções obtidas [46].

O uso da R-Metric é mais adequado ao problema descrito, pois outras métricas tradicionais de avaliação de desempenho não levam em consideração o RP informado pelo DM durante o processo de avaliação. A Figura 6.1 mostra o fluxograma do cálculo da R-Metric.

Em resumo, são realizados cinco passos para o cálculo da R-Metric. O primeiro passo (Pré-processamento) consiste em filtrar as soluções, mantendo apenas as soluções não-dominadas e não repetidas. O segundo passo (Identificação do Pivô) é responsável pela identificação de um ponto representativo que reflete a satisfação geral das soluções em relação ao RP fornecido pelo DM, para o qual a solução mais próxima é usada em relação ao RP.

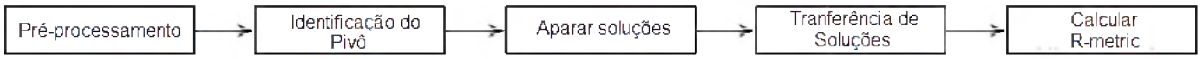



Figura 6.1: Fluxograma da R-Metric (adaptada de [46]).

Na terceira etapa (Aparar soluções), o objetivo principal é ter uma comparação justa entre diferentes conjuntos, cortando pontos que estão fora da ROI. Embora o procedimento de otimização seja esperado para executar determinada tarefa, essa etapa está incluída para garantir que apenas as soluções dentro de uma distância euclidiana igual a $\frac{\Delta}{2}$ relativas ao pivô sejam mantidas, ou seja, apenas as soluções localizadas nesta ROI aproximada são válidas para avaliação de desempenho. Com isso, cada algoritmo forma a ROI com base em suas soluções e a ROI acaba não sendo considerada como uma região fixa no espaço de busca.

O quarto estágio (Transferência de Soluções) é considerado o núcleo da R-Metric, onde os pontos cortados são transferidos para uma posição virtual. Então, é possível avaliar sua proximidade com a ROI ao longo da direção preferida. Finalmente, o último passo (Calcular R-Métric) consiste em aplicar o indicador de qualidade nas soluções processadas pela R-Metric. Neste trabalho, utiliza-se o indicador de qualidade hipervolume [93].

O hipervolume mede a área de cobertura que uma fronteira de Pareto conhecida ($PF_{known}$) exerce sobre o espaço objetivo. A Equação 6.1 apresenta o cálculo do hipervolume.

$$HV = \left( \bigcup_i vol_i : vec_i \in PF_{known} \right) \tag{6.1}$$

De maneira geral, o valor do hipervolume (HV) é dado pela união dos volumes ($vol$) de vetores de valores objetivos ($vec_i$) de soluções $i$ presentes na fronteira de Pareto $PF_{known}$. Em um problema de apenas dois objetivos, o volume de um vetor de objetivos de uma solução $i$ é na verdade a área retangular ligada pelos pontos ($z_1(i)$, $z_2(i)$) e por um ponto de referência (não confundir com o RP informado pelo DM).

O objetivo do hipervolume com a R-Metric (R-HV) é avaliar a disseminação de soluções pela ROI e, ao mesmo tempo, a proximidade dessas soluções com o RP. O R-HV foi calculado considerando os conjuntos $PF_{aprox}$ gerados por cada algoritmo. No final, retorna-se a média dos resultados obtidos ao calcular o R-HV em cada conjunto. Para isso, os valores dos objetivos foram normalizados entre [0.0; 1.01].

### Distância Euclidiana (DE)

Este indicador de qualidade mede a distância entre uma solução qualquer e uma solução ideal em um espaço de soluções. Considerando a situação em que o DM escolherá apenas uma solução do conjunto não dominado, analisam-se as soluções com os menores valores de distância euclidiana em relação ao RP informado pelo DM. Assim, o objetivo é medir a proximidade de uma determinada solução com o RP.

Para executar o cálculo de DE, utilizou-se o conjunto $PF_{known}$ de cada algoritmo. A Equação 6.2 descreve a fórmula para calcular o valor referente a distância entre dois pontos $P = \{p_1, p_2, ..., p_n\}$ e $Q = \{q_1, q_2, ...,q_n\}$ em um espaço euclidiano n-dimensional.

$$DE = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{6.2}$$

**Número médio de soluções na ROI**

Esta métrica visa avaliar o quão bem um algoritmo pode gerar soluções na ROI. Para complementar a análise desta métrica, também consideram-se a porcentagem de soluções geradas na ROI em relação a todas as soluções geradas. A Equação 6.3 descreve a fórmula para calcular a porcentagem de soluções na ROI.

$$P_{roi} = \frac{|S_{ROI}|}{|S|} \times 100 \tag{6.3}$$

onde $S_{ROI}$ representa o conjunto de soluções que estão dentro da ROI e $S$ é o conjunto formado por todas as soluções obtidas pelo algoritmo. A métrica avalia o desempenho do algoritmo no sentido da capacidade de obter soluções concentradas que satisfaçam as preferências do DM. Para realizar o cálculo de ambas as métricas, a média foi calculada considerando os conjuntos $PF_{aprox}$ formados por cada algoritmo e para a determinação da ROI, foi utilizada a R-Metric descrita anteriormente.

Normalmente, em SBSE, os pesquisadores usam testes não-paramétricos, levando em consideração a estocasticidade de algoritmos baseados em busca. Nesse sentido, para comparar os algoritmos para cada problema/instância, foi usado o teste estatístico de Kruskal-Wallis [50] com 95% de nível de significância.

## 6.1.2 Modelos de Características Utilizados

Foram utilizados quatro FMs já utilizados em outros trabalhos da literatura [60, 81, 31]. Sendo eles: a) James [7], um FM para sistema colaborativo *web*; b) CAS (*Car Audio System*) [88], focado na gestão de sistemas de som automotivo; c) WS (*Weather Station*) [28], um FM de sistemas de previsão do tempo; e d) E-Shop [79], um FM relacionado a uma aplicação de comércio eletrônico.

A Tabela 6.1 mostra detalhes sobre cada FM, como o número de produtos ($n_t$), mutantes vivos ($AM$), pares válidos ($P$) e características (Features). O conjunto AM não inclui mutantes equivalentes e mortos por produtos inválidos. Isso permite a geração de um conjunto composto apenas por produtos válidos para o FM em teste.

Tabela 6.1: Características dos FM utilizados no experimento.

| **FM** | *nt* | *AM* | *P* | **Features** |
|---|---|---|---|---|
| James | 68 | 106 | 75 | 14 |
| CAS | 450 | 227 | 183 | 21 |
| WS | 504 | 357 | 195 | 22 |
| E-Shop | 1152 | 394 | 202 | 22 |

## 6.1.3 Definição dos Pontos de Referência

Um passo importante para realizar os experimentos é a definição do Ponto de Referência (RP) que será usado para cada teste. Para o problema deste trabalho, esta tarefa pode ser considerada complexa, uma vez que o DM, em geral, não tem conhecimento sobre a frente de

Pareto do problema. Por isso, foram estabelecidos três RPs que representam possíveis opções que podem ser tomadas para o processo de otimização. A definição desses pontos foi baseada no trabalho desenvolvido por [77]. Os RPs são chamados como Viável, Inviável e Real descritos da seguinte forma.

- **Viável**: é definido considerando a ideia de uma solução que ainda pode ser melhorada pelo processo de otimização;
- **Inviável**: foi definido em oposição ao anterior, leva em consideração uma solução que não pode ser alcançada pelo processo de otimização;
- **Real**: é definido considerando uma solução a uma pequena distância da fronteira formada pelo conjunto $PF_{true}$.

Esses RPs foram escolhidos considerando o conhecimento prévio da fronteira gerada pelo algoritmo NSGA-II obtida em um trabalho da literatura [61]. Além disso, a definição dos pontos leva em conta um intervalo de 95% a 99% de cobertura e um número médio de produtos cujos valores objetivos estavam no intervalo de cobertura desejado. A Tabela 6.2 mostra os RPs usados nos experimentos para as formulações de 2 objetivos e 3 objetivos.

Tabela 6.2: Pontos de referência.

| Experimento | FM | Viável | Inviável | Real |
|---|---|---|---|---|
| 2-Objetivos | James | (6; 95%) | (2; 98%) | (5; 97%) |
| | CAS | (8; 96%) | (3; 97%) | (7; 97%) |
| | WS | (11; 96%) | (3; 97%) | (10; 98%) |
| | E-Shop | (11; 95%) | (3; 97%) | (9; 98) |
| 3-Objetivos | James | (6; 98%; 98%) | (1; 98%; 98%) | (3; 98%; 98%) |
| | CAS | (8; 96%; 96%) | (3; 98%; 98%) | (5; 98%; 99%) |
| | WS | (12; 98%; 98%) | (3; 97%; 97%) | (8; 98%; 99%) |
| | E-Shop | (12; 98%; 98%) | (3; 99%; 99%) | (5; 97%; 98%) |

### 6.1.4 Configuração dos Parâmetros

Antes da execução dos experimentos, realizou-se um ajuste de parâmetros com base em alguns valores já bem definidos em alguns trabalhos da literatura [60, 81, 31]. O *tuning* foi realizado em vez de parâmetros fixos para garantir uma comparação justa considerando os melhores desempenhos dos algoritmos.

As taxas de probabilidade para os operadores genéticos foram igualmente estabelecidas para os três algoritmos, NSGA-II, r-NSGA-II e R-NSGA-II. Para isso, foram considerados os valores utilizados no trabalho desenvolvido por [81], sendo 90% para probabilidade de cruzamento e 0,5% para mutação. Essa utilização é possível porque os algoritmos baseados em preferências são baseados no algoritmo NSGA-II, portanto, bons resultados podem ser obtidos com esses valores.

Em relação à avaliações de *fitness* (neste trabalho considerado como critério de parada) e tamanho da população, foi realizado um ajuste de parâmetros para avaliar diferentes valores para verificar se uma melhor convergência poderia ocorrer entre as soluções dentro da ROI considerando os algoritmos baseados em preferências. Para os algoritmos NSGA-II e NSGA-II-HH, foram adotados os mesmos valores utilizados no trabalho desenvolvido por [81], com 60.000 avaliações de *fitness* e um tamanho de população igual a 200.

Além dos parâmetros mencionados anteriormente, foram avaliados valores diferentes para os parâmetros que controlam a diversidade nos algoritmos baseados em preferências sendo $\epsilon$ no caso de R-NSGA-II e $\delta$ para r-NSGA-II . Para ambos os algoritmos, o peso $w$ utilizado para enfatizar cada objetivo no vetor dos níveis de aspiração (RP) foi o mesmo, sendo $w = 0{,}5$.

Os parâmetros específicos da HH foram definidos com base no trabalho de [32] e [35], sendo o parâmetro $c$ foi definido com 5.0, a janela de tempo foi definida para 25% do tamanho da população, para o FRR-MAB. Já para o CF foram definidos $\alpha = 1.0$ e $\beta = 0.00005$. Para R-NSGA-II-HH e r-NSGA-II-HH foi utilizada uma taxa de mutação e cruzamento de 100%. Isso foi definido por causa da característica determinística de ambas as funções de seleção (CF e MAB). Se outros valores fossem usados, as heurísticas de baixo nível poderiam ser avaliadas pelas funções de seleção sem serem executadas se a probabilidade não fosse alcançada. Ao configurar as probabilidades para 100%, assegura-se que as heurísticas de baixo nível sejam executadas e avaliadas pelo seu desempenho recente. Pode-se pensar que 100% para a probabilidade de mutação é um valor muito alto, no entanto, as funções de seleção determinam quando a mutação deve ou não deve ser usada e, respectivamente, selecionar uma heurística de baixo nível com um dos operadores de mutação ou uma heurística de baixo nível contendo apenas um operador de cruzamento. Esta é uma vantagem de usar hiper-heurísticas.

Para realizar a execução dos algoritmos baseados em preferências com cada combinação de parâmetros, o RP inviável foi usado, uma vez que é uma solução que dificilmente pode ser melhorada. Espera-se que os algoritmos utilizados com os melhores valores de parâmetros para este ponto, também serão eficientes para os demais pontos de referência. A Tabela 6.3 exibe os valores avaliados para cada parâmetro durante a fase de ajuste. Para tanto, foram realizadas 10 execuções independentes para cada algoritmo.

Tabela 6.3: Variação de parâmetros.

| **Parâmetro** | **Value** |
|---|---|
| Tamanho população | 50; 100; 200 |
| Max Avaliações | 30,000; 60,000; 90,000 |
| $\delta$ | 0.1; 0.03; 0.05 |
| $\epsilon$ | 0.0001; 0.001; 0.01 |

Após o *tuning*, as melhores configurações de parâmetros foram selecionadas com base nos melhores valores médios de R-HV, considerando o teste estatístico de de Kruskal Wallis [50] com 95% de significância. Caso não ocorra diferença estatística, a configuração com o menor tempo médio de execução foi selecionada. Para todos os algoritmos, os valores para os parâmetros $\delta$ e $\epsilon$ foram os mesmos para todos os FMs e para ambas as formulações, sendo respectivamente 0,3 e 0,001.

A Tabela A.1 (Apêndice A) mostra as melhores configurações para o tamanho da população e para número máximo de avaliações consideradas para cada FM/algoritmo, considerando os algoritmos r-NSGA-II e R-NSGA-II. As Tabelas A.2 e A.3 apresentam os parâmetros

para os algoritmos r-NSGA-II-HH e R-NSGA-II-HH. Na Tabela A.2, os algoritmos intitulados r-NSGA-II-HH-CF representam a HH com o método Choice Function, r-NSGA-II-HH-FRR representa o FRR-MAB e r-NSGA-II-HH-R representa a HH com o método aleatório. A Tabela A.3 segue a mesma lógica, porém para o algoritmo R-NSGA-II-HH.

Para a fase experimental, descrita na próxima seção, foram realizadas 30 execuções independentes para cada algoritmo, considerando os parâmetros estabelecidos nesta seção.

## 6.2 RESULTADOS E DISCUSSÕES

Nesta seção, os resultados experimentais são apresentados e analisados com o objetivo de responder às questões de pesquisa propostas. Buscando facilitar a compreensão, as análises e respostas para as respectivas questões de pesquisa são apresentadas em subseções distintas.

A Subseção 6.2.1 apresenta a estrutura geral dos resultados, com todas as tabelas contendo cada resultado experimental e a análise destes resultados para responder a referida questão de pesquisa. Para responder as demais questões de pesquisa, devido à semelhança das análises realizadas nos experimentos, as tabelas completas com as informações detalhadas de cada experimento, encontram-se nos apêndices B, C, D. Tal estruturação, foi realizada visando uma melhor leitura e entendimento do resultados.

### 6.2.1 QP-1: PEMOAs x NSGA-II

Nesta subseção são apresentados os resultados da comparação entre os dois algoritmos baseados em preferência, r-NSGA-II e R-NSGA-II com o EMOA tradicional NSGA-II. Os resultados são divididos em duas subseções diferentes, uma para cada formulação de objetivos. Por fim, é realizada uma análise visando a responder a questão de pequisa.

#### Experimento com 2 Objetivos

A Tabela 6.4 apresenta os valores médios e o desvio padrão para o indicador R-HV. Os valores em negrito representam os melhores resultados e as células em cinza claro representam os valores estatisticamente equivalentes, levando em consideração o teste estatístico de Kruskal-Wallis com um nível de confiança de 95%.

Tabela 6.4: R-HV para o experimento com 2-Objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | 0.61416 (0.03940) | **0.82218 (0.07059)** | 0.80574 (0.13429) |
| | Viável | 0.87271 (0.00661) | **0.89414 (0.00725)** | 0.88408 (0.01887) |
| | Real | 0.89690 (0.00561) | 0.89050 (0.01214) | **0.90897 (0.02390)** |
| CAS | Inviável | 0.78761 (0.07616) | 0.90387 (0.04023) | **0.92735 (0.06315)** |
| | Viável | 0.94377 (0.00108) | 0.94740 (0.00369) | **0.96230 (0.02065)** |
| | Real | 0.93657 (0.01110) | 0.94524 (0.00696) | **0.96546 (0.01250)** |
| WS | Inviável | 0.74866 (0.05871) | 0.90380 (0.02607) | **0.94218 (0.02074)** |
| | Viável | 0.93973 (0.00049) | 0.94446 (0.00248) | **0.95950 (0.01287)** |
| | Real | 0.95395 (0.00682) | 0.95918 (0.00301) | **0.96148 (0.01199)** |
| E-Shop | Inviável | 0.93828 (0.05035) | 0.94600 (0.02934) | **0.96452 (0.01328)** |
| | Viável | 0.93844 (0.00945) | 0.94585 (0.00472) | **0.95534 (0.01486)** |
| | Real | 0.94499 (0.02559) | **0.96874 (0.00667)** | 0.96650 (0.02758) |

Na Tabela 6.4 é observado um desempenho melhor dos PEMOAs em todos os casos. O algoritmo R-NSGA-II supera o r-NSGA-II na maioria dos casos independentemente dos RPs adotados (9 dos 12 casos). Também pode-se observar que o r-NSGA-II obteve um melhor desempenho no FM James e também em outros dois casos, quando o RP está próximo ao conjunto $PF_{true}$. A determinação deste ponto na prática é difícil, tendo em vista que em muitos casos, o testador não tem conhecimento sobre a fronteira de Pareto.

A Tabela 6.5 apresenta os resultados da métrica distância euclidiana. Na tabela estão representados o valor de DE da solução mais próxima ao ponto de referência e os valores dos objetivos correspondentes a esta solução. Os valores em negrito representam os melhores resultados.

Tabela 6.5: DE para o experimento com 2 Objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | **0.19752 (3;78.3)** | **0.19752 (3;78.3)** | **0.19752 (3;78.3)** |
| | Viável | 0.05000 (6;100.0) | **0.04056 (6;99.0)** | 0.05000 (6;100.0) |
| | Real | **0.01113 (5;98.1)** | **0.01113 (5;98.1)** | **0.01113 (5;98.1)** |
| CAS | Inviável | 0.11980 (4;85.0) | **0.06266 (5;90.7)** | 0.10225 (5;86.7) |
| | Viável | 0.03118 (8;99.1) | 0.03559 (8;99.5) | **0.00525 (7;96.4)** |
| | Real | 0.00797 (7;97.7) | **0.00083 (7;96.9)** | 0.00356 (7;97.3) |
| WS | Inviável | 0.13252 (5;83.7) | **0.04355 (7;92.7)** | 0.04907 (7;92.1) |
| | Viável | 0.04004 (12;100.0) | **0.02327 (10;98.3)** | 0.04004 (12;100.0) |
| | Real | 0.00879 (10;98.8) | 0.00599 (10;98.5) | **0.00240 (10;97.7)** |
| E-Shop | Inviável | 0.07667 (7;89.3) | 0.01100 (9;97.9) | **0.00782 (12;96.9)** |
| | Viável | 0.04747 (12;99.7) | 0.03735 (9;98.7) | **0.02974 (13;97.9)** |
| | Real | 0.05380 (9;97.4) | **0.01049 (8;96.9)** | 0.03087 (12;94.9) |

A tabela mostra que para o menor FM em relação ao número de produtos (James), os algoritmos têm um desempenho similar. No entanto, para outros casos, pode-se ver que os algoritmos r-NSGA-II e R-NSGA-II são melhores do que o algoritmo tradicional. No entanto, não é possível indicar o melhor PEMOA. Observa-se novamente um melhor desempenho do r-NSGA-II ao usar RPs reais.

Quanto ao número de soluções encontradas, a Tabela 6.6 apresenta o número médio de soluções na ROI e a taxa entre o número de soluções encontradas na ROI e o número total de soluções encontradas. Os valores em negrito representam o menor número de soluções na ROI.

Em geral, o algoritmo r-NSGA-II gera o menor número de soluções e a maioria delas está 100% na ROI. Isso não ocorre em apenas dois casos, onde são usados RPs inviáveis. Mas neste caso, em média 96% das soluções estão na ROI. Por outro lado, é possível ver que o NSGA-II gera um número maior de soluções que não são interessantes para o testador, em média, usando RPs inviáveis apenas 26% das soluções estão dentro da ROI. Ao usar os outros dois tipos de RPs, essa porcentagem é de apenas 50%, o que significa que, nos melhores casos, 50% das soluções geradas pelo NSGA-II não são interessantes do ponto de vista do testador.

O R-NSGA-II pode gerar os melhores *trade-offs*, ou seja, ele gera mais soluções e a maioria delas está dentro da ROI. Considerando os RPs inviáveis, viáveis e reais, a média de porcentagem de soluções dentro da ROI é respectivamente de 77,85%, 83,03% e 85,11%. Observa-se também que o desempenho de todos os algoritmos é melhor quando os RPs próximos da $PF_{true}$ são usados e pior quando são utilizados RPs inviáveis.

Resumindo os resultados para a formulação de 2 objetivos, é possível notar que, ambos os PEMOAs são considerados uma melhor opção em relação ao NSGA-II considerando todos

Tabela 6.6: Número de soluções na ROI para formulação de 2 objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | 1.0 (16.58%) | **1.0 (100%)** | 1.6 (54.11%) |
| | Viável | 2.1 (33.65%) | **1.0 (100%)** | 2.3 (56.83%) |
| | Real | 2.5 (41.98%) | **1.0 (100%)** | 2.3 (65.38%) |
| CAS | Inviável | **1.6 (17.51%)** | 1.7 (92.77%) | 3.0 (71.61%) |
| | Viável | 3.8 (51.29%) | **1.0 (100%)** | 3.9 (81.38%) |
| | Real | 4.6 (60.20%) | **1.3 (100%)** | 4.2 (83.96%) |
| WS | Inviável | **1.4 (11.90%)** | 1.6 (100%) | 3.1 (87.71%) |
| | Viável | 5.2 (46.47%) | **1.1 (100%)** | 3.8 (95.27%) |
| | Real | 5.6 (51.39%) | **1.0 (100%)** | 3.4 (97.52%) |
| E-Shop | Inviável | 4.0 (61.53%) | **2.0 (92.77%)** | 3.8 (98.00%) |
| | Viável | 5.5 (81.49%) | **1.2 (100%)** | 4.1 (98.66%) |
| | Real | 5.7 (84.81%) | **1.9 (100%)** | 4.4 (93.61%) |
| **Média** | Inviável | 2.0 (26.88%) | 1.6 (96.38%) | 2.8 (77.85%) |
| | Viável | 4.5 (53.22%) | 1.1 (100%) | 3.5 (83.03%) |
| | Real | 4.6 (59.59%) | 1.3 (100%) | 3.5 (85.11%) |

os indicadores. O algoritmo R-NSGA-II supera os demais na maioria das instâncias e RPs considerando o indicador R-HV. As soluções obtidas pelo R-NSGA-II têm mais diversidade e convergência quando comparadas com o r-NSGA-II. Considerando o número de soluções na ROI, o R-NSGA-II apresenta o melhor *trade-off* entre o número de soluções interessantes e não interessantes. Além disso, se o testador estiver interessado em visualizar poucas soluções, o r-NSGA-II parece ser uma boa opção, tendo em vista que na maioria dos casos, 100% das soluções estão na ROI.

### Experimento com 3 Objetivos

A Tabela 6.7 apresenta os valores médios e os desvios-padrão para o indicador R-HV. Os valores em negrito representam os melhores e as células em cinza claro representam valores estatisticamente equivalentes.

Tabela 6.7: R-HV para o experimento com 3 Objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | 0.49610 (0.01141) | 0.57369 (0.13617) | **0.72405 (0.15581)** |
| | Viável | 0.89049 (0.00097) | 0.85969 (0.05761) | **0.88577 (0.03545)** |
| | Real | 0.78968 (0.01483) | 0.77076 (0.04495) | **0.81724 (0.03200)** |
| CAS | Inviável | 0.70828 (0.08066) | **0.83156 (0.04331)** | 0.81617 (0.08614) |
| | Viável | **0.93593 (0.01714)** | 0.88971 (0.03290) | 0.90847 (0.05583) |
| | Real | 0.84741 (0.04599) | **0.89569 (0.04119)** | 0.88627 (0.05863) |
| WS | Inviável | 0.65710 (0.07579) | 0.87406 (0.03200) | **0.91703 (0.03360)** |
| | Viável | 0.94546 (0.00354) | 0.94266 (0.00738) | **0.95499 (0.01533)** |
| | Real | 0.91904 (0.01920) | 0.94076 (0.00718) | **0.95438 (0.01698)** |
| E-Shop | Inviável | 0.94512 (0.01691) | 0.94612 (0.01388) | **0.95677 (0.02156)** |
| | Viável | 0.96778 (0.01004) | 0.93575 (0.01525) | **0.96979 (0.01149)** |
| | Real | 0.93755 (0.02548) | 0.93225 (0.01381) | **0.96046 (0.02258)** |

Novamente, o algoritmo R-NSGA-II gera os melhores resultados na maioria dos FMs e RPs (9 em 12 casos). Em apenas um caso (no FM CAS), o tradicional NSGA-II gera o melhor resultado, mas sem diferença estatística com R-NSGA-II. Com relação ao r-NSGA-II, ele gera os melhores resultados em dois casos, na instância CAS. É notável que o melhor desempenho do NSGA-II é obtido usando RPs viáveis.

A Tabela 6.8 apresenta os resultados de DE da solução mais próxima ao ponto de referência e os valores dos objetivos que formulam está solução. A tabela mostra que, apenas para o menor FM em relação ao número de produtos (James) e RPs reais, os algoritmos têm uma desempenho similar. Diferente da formulação de 2 objetivos anteriormente descrita, o r-NSGA-II gerou melhores resultados em comparação com os outros na maioria dos FMs e RPs (10 em 12). É importante notar que o algoritmo NSGA-II gera bons resultados quando são utilizados RPs reais para as instâncias James e E-Shop.

Tabela 6.8: DE para o experimento com 3 objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | 0.22729 (3;75;98) | **0.20016 (3;78;96)** | **0.20016 (3;78;96)** |
| | Viável | 0.02828 (6;100;100) | **0.02485 (5;98;100)** | 0.02828 (6;100;100) |
| | Real | **0.06077 (4;92;100)** | **0.06077 (4;92;100)** | **0.06077 (4;92;100)** |
| CAS | Inviável | 0.14769 (4;83;98) | **0.03264 (6;95;99)** | 0.18762 (4;79;99) |
| | Viável | 0.05661 (9;100;100) | **0.04229 (7;97;100)** | 0.05661 (9;100;100) |
| | Real | 0.09506 (5;88;100) | **0.04720 (6;93;100)** | 0.08630 (5;89;100) |
| WS | Inviável | 0.16337 (5;82;89) | 0.05573 (7;92;94) | **0.02345 (8;94;96)** |
| | Viável | 0.02828 (12;100;100) | **0.01210 (10;98;98)** | 0.02828 (12;100;100) |
| | Real | 0.01477 (9;96;98) | **0.00985 (9;97;98)** | 0.07464 (7;92;94) |
| E-Shop | Inviável | 0.04378 (8;94;98) | **0.02003 (23;98;100)** | 0.07491 (10;93;94) |
| | Viável | 0.01553 (9;97;99) | **0.00987 (20;98;98)** | 0.04208 (10;93;97) |
| | Real | **0.02400 (8;94;98)** | 0.03723 (16;93;98) | 0.03289 (14;99;100) |

Quanto ao número de soluções encontradas, a Tabela 6.9 mostra o número médio de soluções na ROI e proporção entre o número de soluções encontradas na ROI e o número total de soluções encontradas. Os valores em negrito representam o menor número de soluções na ROI.

Da mesma forma que aconteceu na formulação de 2 objetivos, em todos os casos, o algoritmo r-NSGA-II gera, em média, o menor número de soluções, mas agora com 100% na ROI, ou seja, todas as soluções. Por outro lado, o R-NSGA-II ainda pode gerar o melhor *trade-off*, uma vez que gera proporcionalmente mais soluções na ROI, ou seja, mais soluções e a maioria delas está dentro da ROI (em média 92,8%, 93,5% e 93,1% para os RPs inviáveis, viáveis e reais, respectivamente). É interessante observar que a porcentagem média das soluções na ROI dos PEMOAs melhorou em relação à formulação de 2 objetivos. O algoritmo NSGA-II gerou cerca de 40% de soluções não interessantes.

Resumindo os resultados para a formulação de 3 objetivos, é possível notar que o desempenho permanece semelhante quando comparado à formulação de 2 objetivos. Os PEMOAs são melhores que o tradicional NSGA-II. O algoritmo R-NSGA-II supera os outros na maioria dos FMs e RPs em relação ao R-HV, que representa a diversidade, convergência e proximidade das soluções com o RP. Mas para esta formulação, o r-NSGA-II apresentou melhores resultados considerando o indicador DE. Com relação ao número de soluções na ROI, novamente, r-NSGA-II gera números menores e todas as soluções geradas estão dentro da ROI. O R-NSGA-II apresentou os melhores *trade-offs*, e deve ser usado se o testador quiser visualizar mais soluções dentro da ROI. No entanto, se o testador estiver interessado em selecionar apenas uma solução

Tabela 6.9: Número de soluções na ROI para formulação de 3 objetivos.

| FM | RP | NSGA-II | r-NSGA-II | R-NSGA-II |
|---|---|---|---|---|
| James | Inviável | 2.7 (72.46%) | **1.8 (100%)** | 2.4 (81.83%) |
| | Viável | 2.1 (35.72%) | **1.0 (100%)** | 2.5 (85.44%) |
| | Real | 1.8 (62.95%) | **1.3 (100%)** | 2.7 (89.78%) |
| CAS | Inviável | 5.2 (76.2%) | **2.2 (100%)** | 8.1 (90.56%) |
| | Viável | 6.4 (65.7%) | **1.3 (100%)** | 6.5 (90.56%) |
| | Real | 6.1 (59.28%) | **2.0 (100%)** | 7.3 (86.24%) |
| WS | Inviável | 5.9 (82.87%) | **3.4 (100%)** | 3.7 (100%) |
| | Viável | 6.8 (74.74%) | **1.2 (100%)** | 3.9 (99.39%) |
| | Real | 10.9 (59.17%) | **2.0 (100%)** | 3.6 (99.17%) |
| E-Shop | Inviável | 4.0 (85.86%) | **3.1 (100%)** | 7.8 (98.82%) |
| | Viável | 4.5 (82.1%) | **3.0 (100%)** | 7.8 (98.77%) |
| | Real | 3.7 (89.98%) | **2.9 (100%)** | 7.7 (97.43%) |
| **Média** | Inviável | 4.4 (79.3%) | 2.1 (100%) | 5.5 (92.8%) |
| | Viável | 4.9 (64.5%) | 1.6 (100%) | 5.1 (93.5%) |
| | Real | 5.6 (67.8%) | 2.0 (100%) | 5.3 (93.1%) |

próxima do RP fornecido, o r-NSGA-II deve ser considerado. O NSGA-II pode ser uma boa opção somente quando RPs reais forem usados, porém na grande maioria dos casos, a fronteira real é desconhecida.

### Considerações gerais

Na grande maioria dos casos, os PEMOAs (r-NSGA-II e R-NSGA II) apresentam melhor desempenho do que o tradicional NSGA-II. Isso é possível porque esses algoritmos têm um mecanismo de busca focado em encontrar boas soluções levando em consideração o RP, enquanto o NSGA-II gera soluções que podem ser consideradas desinteressantes do ponto de vista do testador. Como consequência, os PEMOAs podem gerar mais soluções dentro da ROI para ambas as formulações, independentemente do tipo de RP fornecido. Os resultados das Tabelas 6.6 e 6.9 demonstram essa situação.

Além disso, considerando a situação em que o testador quer ver apenas uma ou poucas soluções, os PEMOAs continuam gerando os melhores resultados porque suas soluções são as mais próximas do RP. Finalmente, os PEMOAs podem gerar menos soluções do que o tradicional NSGA-II. Neste contexto, esse fato é importante porque isso pode reduzir o esforço do testador na tarefa de selecionar a melhor solução.

Para ilustrar esta vantagem, a Tabela 6.10 apresenta o número de soluções geradas por cada algoritmo dentro da ROI e o número fora da ROI (não interessantes), para o FM WS usando RPs inviáveis, considerando o conjunto $PF_{known}$ e a formulação de 2 objetivos. O NSGA-II gera 12 possíveis soluções, mas apenas 4 (33%) estão dentro da ROI, quase 67% delas não são interessantes do ponto de vista do testador. Por outro lado, o R-NSGA-II gera 4 (80%) soluções dentro da ROI, enquanto o r-NSGA-II gera 3 soluções possíveis (100%) dentro da ROI. Nesse sentido, o uso dos PEMOAs é vantajoso.

De maneira geral, o algoritmo R-NSGA-II pode ser considerado como melhor opção se o DM estiver interessado em selecionar ou visualizar mais soluções próximas ao RP, ou seja, pode gerar mais soluções dentro da ROI. Essas descobertas são facilmente demonstradas quando são levados em consideração os valores de R-HV e a porcentagem de soluções dentro da ROI. Pode-se observar que o desempenho do r-NSGA-II melhora quando o número de objetivos

Tabela 6.10: Número de soluções por algoritmo considerando WS e o RP inviável.

| Algoritmo | Solução na ROI | Não interessantes |
|---|---|---|
| **NSGA-II** | 4 | 8 |
| **r-NSGA-II** | 3 | 0 |
| **R-NSGA-II** | 4 | 1 |

aumenta, e se o testador fornecer RPs próximos ao conjunto $PF_{true}$, mas isso deve ser investigado em experimentos futuros.

No entanto, se o testador estiver interessado na seleção de apenas algumas soluções, o r-NSGA-II deve ser considerado porque gera o menor número de soluções na ROI e, na maioria dos casos, não gera nenhuma solução fora dela. Esse comportamento acontece em ambas as formulações.

### 6.2.2 QP-2: FRR-MAB x CF x Aleatório

Os resultados dessa subseção visam a responder a QP-2 e comparar os métodos de seleção a serem usados com os algoritmos R-NSGA-II-HH e r-NSGA-II-HH. Assim como na subseção anterior, os resultados de ambas as formulações são apresentados separadamente. As tabelas de R-HV, DE e número de soluções na ROI são apresentadas no Apêndice B.

A Tabela 6.11 apresenta o número de casos nos quais cada método, com o algoritmo r-NSGA-II-HH, obteve os melhores resultados, considerando os indicadores R-HV e DE com as formulações experimentais com 2 e 3 objetivos. Esses valores são obtidos das Tabelas B.1 e B.7 referentes ao indicador R-HV e das Tabelas B.3 e B.9 referentes ao indicar DE. Na tabela r-NSGA-II-HH-CF representa o algoritmo com o método de seleção Choice Function, r-NSGA-II-HH-FRR representa o algoritmo com o método FRR-MAB e por fim, r-NSGA-II-HH-R representa algoritmo com o método aleatório.

Tabela 6.11: Número de casos nos quais os métodos de seleção, considerando o algoritmo r-NSGA-II-HH, obtêm os melhores resultados de R-HV e DE.

| Indicador | Algoritmo | F2O | F3O |
|---|---|---|---|
| R-HV | r-NSGA-II-HH-CF | 7(12) | 11(12) |
| | r-NSGA-II-HH-FRR | 11(12) | 12(12) |
| | r-NSGA-II-HH-R | 12(12) | 12(12) |
| DE | r-NSGA-II-HH-CF | 7(12) | 6(12) |
| | r-NSGA-II-HH-FRR | 8(12) | 4(12) |
| | r-NSGA-II-HH-R | 7(12) | 8(12) |

Considerando o experimento com 2 objetivos, observa-se que o algoritmo r-NSGA-II-HH com o método de seleção aleatório é melhor considerando R-HV e o método r-NSGA-II-HH-FRR é melhor para o valor de ED. Considerando 3 objetivos essa diferença não existe e o aleatório é melhor em valores de ED.

A Tabela 6.12 apresenta as médias de número de soluções e porcentagem de soluções na ROI para cada algoritmo r-NSGA-II-HH, considerando as formulações com 2 e 3 objetivos.

De maneira geral, na formulação com 2 objetivos, é destacado que o algoritmo r-NSGA-II-HH-R consegue obter melhores resultados considerando as médias de R-HV e também foi o que gerou um menor número de soluções dentro da ROI, entretanto, o algoritmo r-NSGA-II-HH-FRR consegue produzir a solução mais próxima aos RPs na maioria das vezes, além disso, ainda foi o que gerou o maior número de soluções dentro da ROI.

Para o experimento com 3 objetivos, os algoritmos r-NSGA-II-HH-FRR e r-NSGA-II-HH-R, mostram-se muito semelhantes. Aparentemente, o algoritmo r-NSGA-II-HH-R produz soluções mais próximas ao RP do que o r-NSGA-II-HH-FRR. Porém, o r-NSGA-II-HH-FRR apresenta um maior número de soluções na ROI e ao mesmo tempo todas sempre na ROI (100%), o que não ocorre no r-NSGA-II-HH-R.

Tabela 6.12: Número e porcentagem da média de soluções na ROI com r-NSGA-II-HH.

| Formulação | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| 2 Objetivos | Inviável | 1.6 (99.44%) | 1.7 (99.44%) | 1.3 (99.16%) |
| | Viável | 1.9 (100%) | 2.1 (100%) | 1.9 (100%) |
| | Real | 1.4 (100%) | 1.5 (100%) | 1.3 (100%) |
| 3 Objetivos | Inviável | 2,4 (100%) | 2,5 (100%) | 2,7 (100%) |
| | Viável | 2,3 (100%) | 2,2 (100%) | 2,2 (100%) |
| | PFTrue | 2,4 (100%) | 2,6 (100%) | 2,2 (99,84%) |

A Tabela 6.13 apresenta o número de casos nos quais cada método, em combinação com o algoritmo R-NSGA-II-HH, obteve os melhores resultados, considerando os indicadores R-HV e DE com as formulações experimentais com 2 e 3 objetivos. Esses valores são obtidos das Tabelas B.2 e B.8 referentes ao indicador R-HV e das Tabelas B.4 e B.10 referentes ao indicar DE.

Tabela 6.13: Número de casos nos quais os métodos de seleção, com o algoritmo R-NSGA-II-HH, obtêm os melhores resultados de R-HV e DE.

| Indicador | Algoritmo | F2O | F3O |
|---|---|---|---|
| R-HV | R-NSGA-II-CF | 0(12) | 0(12) |
| | R-NSGA-II-FRR | 12(12) | 12(12) |
| | R-NSGA-II-R | 12(12) | 11(12) |
| DE | R-NSGA-II-CF | 3(12) | 5(12) |
| | R-NSGA-II-FRR | 10(12) | 5(12) |
| | R-NSGA-II-R | 3(12) | 5(12) |

A Tabela 6.14 apresenta as médias de número de soluções e porcentagem de soluções na ROI para cada algoritmo R-NSGA-II-HH, considerando a formulação com 2 e 3 objtivos.

Tabela 6.14: Número e porcentagem da média de soluções na ROI com R-NSGA-II-HH.

| Formulação | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| 2 Objetivos | Inviável | 2.5 (63.30%) | 4.7 (49.2%) | 4.9 (47.8%) |
| | Viável | 3.3 (74.71%) | 5.1 (73.1%) | 5.2 (71.18%) |
| | Real | 3.2 (77.3%) | 5.4 (71.1%) | 5.5 (68.25%) |
| 3 Objetivos | Inviável | 9.1 (91.76%) | 10.5 (88.89%) | 9.2 (90.48%) |
| | Viável | 6.8 (78.78%) | 9.2 (79.34%) | 6.9 (79.86%) |
| | Real | 8.2 (88.57%) | 9.9(86.5%) | 8.1 (86.93%) |

Considerando a formulação com 2 objetivos, para o algoritmo R-NSGA-II-HH ocorre uma conclusão difícil em relação a cada método de seleção. O algoritmo R-NSGA-II-HH-R produz melhores valores de R-HV ou estatisticamente equivalentes ao melhor, em todos os casos, e ao mesmo tempo o algoritmo R-NSGA-II-HH-FRR é estatisticamente equivalente ou melhor na grande maioria dos casos. Porém, o algoritmo R-NSGA-II-HH-FRR mostra-se como uma

melhor opção quando o testador deseja encontrar uma única solução em relação ao RP, tendo em vista que este, apresentou os melhores resultados para esta comparação na grande maioria dos casos. Já em relação à quantidade de soluções na ROI, o algoritmo R-NSGA-II-HH-CF, produz o menor número de soluções já o algoritmo R-NSGA-II-HH-R produz um maior número de soluções na ROI, tendo assim, uma maior diversidade de soluções na ROI, embora produza mais soluções fora da ROI em relação ao R-NSGA-II-HH-FRR, com uma diferença muito pequena em relação a média de soluções.

Já na formulação com 3 objetivos, é possível notar que o algoritmo R-NSGA-II-HH-FRR, tem uma prevalência de melhores resultados e produz um melhor *trade-off* dentro da ROI. Além disso, os três algoritmos obtiveram resultados semelhantes em relação a quem gera a solução mais próxima aos RPs, porém, os algoritmos R-NSGA-II-HH-CF e R-NSGA-II-HH-FRR geram soluções mais próximas quando consideram-se as maiores instâncias.

De maneira geral, tornou-se difícil definir qual dos métodos de seleção foi o melhor, quando aplicado em cada algoritmo, r-NSGA-II-HH e R-NSGA-II-HH, respectivamente. Pois, cada método se mostrou melhor em um caso específico como descrito anteriormente. Entretanto, pode-se observar em um maior número de casos, nos quais os melhores resultados encontrados foram originados pelo método FRR-MAB com ambos os algoritmos r-NSGA-II-HH e R-NSGA-II-HH. Este método foi utilizado para responder as próximas questões de pesquisa.

### 6.2.3 QP-3: HH x PEMOAs

Nesta seção, são apresentados os resultados da comparação dos algoritmos com o método de seleção escolhido na QP-2 (r-NSGA-II-HH-FRR e R-NSGA-II-HH-FRR) com os PEMOAs tradicionais. As tabelas de R-HV, DE e número de soluções na ROI são apresentadas no Apêndice C.

A Tabela 6.15 apresenta o número de casos nos quais os algoritmos apresentaram os melhores resultados, considerando o indicador R-HV e as formulações experimentais com 2 e 3 objetivos. Esses valores são obtidos das Tabelas C.1 e C.4. Da mesma forma, a Tabela 6.16 apresenta o número de casos para o indicador DE, sendo obtidos das Tabela C.2 e C.5.

Tabela 6.15: Número de casos nos quais os PEMOAs e HH obtiveram os melhores resultados de R-HV.

| Algoritmo | F2O | F3O |
|---|---|---|
| r-NSGA-II | 1(12) | 0(12) |
| r-NSGA-II-HH-FRR | 6(12) | 7(12) |
| R-NSGA-II | 8(12) | 7(12) |
| R-NSGA-II-HH-FRR | 10(12) | 11(12) |

Tabela 6.16: Número de casos nos quais os PEMOAs e HH obtiveram os melhores resultados de DE.

| Algoritmo | F2O | F3O |
|---|---|---|
| r-NSGA-II | 3(12) | 4(12) |
| r-NSGA-II-HH-FRR | 6(12) | 6(12) |
| R-NSGA-II | 3(12) | 1(12) |
| R-NSGA-II-HH-FRR | 2(12) | 4(12) |

De maneira geral, pode-se observar nos experimentos que as HHs obtiveram melhores resultados na grande maioria dos casos em relação aos PEMOAs. Além disso, em ambas as

formulações o algoritmo R-NSGA-II-HH-FRR apresenta melhores valores de R-HV e também um melhor *trade-off* de soluções na ROI em relação ao algoritmo r-NSGA-II-HH-FRR. A Tabela 6.17 apresenta as médias do número de soluções e a porcentagem de soluções na ROI obtidas por cada algoritmo.

Tabela 6.17: Número e porcentagem de soluções na ROI entre HH e PEMOA.

| Formulação | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| 2 Objetivos | Inviável | 1.6 (96.38%) | 1.7 (99.44%) | 2.8 (77.85%) | 4.7 (49.2%) |
| | Viável | 1.1 (100%) | 2.1 (100%) | 3.5 (83.03%) | 5.1 (73.1%) |
| | Real | 1.3 (100%) | 1.5 (100%) | 3.5 (85.11%) | 5.4 (71.1%) |
| 3 Objetivos | Inviável | 2.1 (100%) | 2,5 (100%) | 5.5 (92.8%) | 10.5 (88.89%) |
| | Viável | 1.6 (100%) | 2,2 (100%) | 5.1 (93.5%) | 9.2 (79.34%) |
| | Real | 2.0 (100%) | 2,6 (100%) | 5.3 (93.1%) | 9.9(86.5%) |

Os resultados obtidos pelos algoritmos R-NSGA-II-HH-FRR e r-NSGA-II-HH-FRR são similares aos resultados obtidos com os seus correspondentes PEMOAs. Ou seja, o algoritmo R-NSGA-II-HH-FRR obtêm resultados superiores e proporciona uma melhor diversidade de soluções dentro do ROI enquanto que o o algoritmo r-NSGA-II-HH-FRR tende a concentrar as soluções mais próximas com uma menor diversidade em relação ao RP.

Pode-se notar que para o algoritmo r-NSGA-II, a sua versão utilizando a HH obteve resultados significativamente melhores em quase todos os experimentos e em ambas as formulações, principalmente considerando o indicador R-HV. Com isso, concluí-se que é mais viável para o testador a utilização do algoritmo r-NSGA-II-FRR em relação ao r-NSGA-II, tendo em vista os melhores resultados obtidos.

Já para o algoritmo R-NSGA-II, sua versão utilizando a HH obteve bons resultados na formulação com 2 objetivos, sendo melhor na grande maioria dos casos. Na formulação com 3 objetivos, esta diferença é um pouco mais discreta. De certa forma, o algoritmo R-NSGA-II-HH obtêm resultados melhores, mas em muitas vezes, o algoritmo R-NSGA-II, se assemelha, ou ainda consegue ser melhor. Este fato é justificado, pois o algoritmo R-NSGA-II já obtém resultados considerados muito bons, dessa forma, a utilização da HH acaba não conseguindo obter grandes melhoras no resultados já obtidos.

Uma possível explicação também é o fato de que o R-NSGA-II-HH-FRR, não aplica o conceito de r-dominância e portanto seu comportamento é igual ao da HH tradicional e talvez por isso não consiga melhorar o desempenho do correspondente R-NSGA-II.

### 6.2.4 QP-4: R-NSGA-II-HH e r-NSGA-II-HH x NSGA-II-HH

Esta subseção apresenta os resultados e análises necessárias para responder a quarta e última questão de pesquisa, que visa a comparar os algoritmos r-NSGA-II-HH-FRR e R-NSGA-II-HH-FRR, com a melhor HH conhecida na literatura já aplicada ao mesmo problema desse trabalho. Está HH será chamada aqui de NSGA-II-HH, refletindo a utilização do algoritmo NSGA-II e método de seleção FRR-MAB [81]. As tabelas de R-HV, DE e número de soluções na ROI são apresentadas no Apêndice D.

A Tabela 6.18 apresenta o número de casos nos quais os algoritmos NSGA-II-HH, r-NSGA-II-HH-FRR e R-NSGA-II-HH-FRR apresentam os melhores resultados ou com igualdade estatística, considerando o indicador R-HV e as formulações experimentais com 2 e 3 objetivos. Esses valores são obtidos das Tabelas D.1 e D.4. Da mesma forma, a Tabela 6.19 apresenta o número de caso do melhor resultado para o indicador DE, sendo obtidos das Tabela D.2 e D.5.

Tabela 6.18: Número de casos nos quais as HHs obtiveram os melhores resultados de R-HV.

| Algoritmo | F2O | F3O |
|---|---|---|
| NSGA-II-HH-FRR | 3(12) | 6(12) |
| r-NSGA-II-HH-FRR | 5(12) | 7(12) |
| R-NSGA-II-HH-FRR | 10(12) | 8(12) |

Tabela 6.19: Número de casos nos quais as HHs obtiveram os melhores resultados de DE.

| Algoritmo | F2O | F3O |
|---|---|---|
| NSGA-II-HH-FRR | 2(12) | 2(12) |
| r-NSGA-II-HH-FRR | 9(12) | 8(12) |
| R-NSGA-II-HH-FRR | 5(12) | 5(12) |

Com base nos resultados mostrados anteriormente, pode-se concluir que ambos os algoritmos HH baseados em preferência obtêm melhores resultados do que o algoritmo NSGA-II-HH. A abordagem pode gerar, na maioria dos casos, os melhores *trade-offs* em relação ao indicador R-HV, soluções mais próximas dos RPs e um maior e menor número de soluções dentro da ROI.

Além disso, a porcentagem de soluções na ROI é um fator que viabiliza a abordagem em relação ao algoritmo NSGA-II-HH. A Tabela 6.20 apresenta as médias de número de soluções e porcentagem de soluções na ROI para cada algoritmo. Na grande maioria dos casos as porcentagens de soluções dentro da ROI encontradas para as HHs baseadas em preferência é superior a 80%, enquanto que o NSGA-II-HH tem em média 70% das soluções fora da ROI. Dessa forma, a utilização do algoritmo NSGA-II-HH dificulta a escolha do testador em determinar a solução que mais satisfaça suas necessidades. Sendo que este fato, acaba não ocorrendo na utilização dos algoritmos r-NSGA-II-HH-FRR e R-NSGA-II-HH-FRR.

Tabela 6.20: Número e porcentagem de soluções na ROI entre NSGA-II-HH, r-NSGA-II-HH e R-NSGA-II-HH.

| Formulação | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| 2 Objetivos | Inviável | 1,1 (12,3%) | 1.7 (99.44%) | 4.7 (49.2%) |
| | Viável | 4,6 (43,9%) | 2.1 (100%) | 5.1 (73.1%) |
| | Real | 4,9 (48%) | 1.5 (100%) | 5.4 (71.1%) |
| 3 Objetivos | Inviável | 5,5 (30,3%) | 2,5 (100%) | 10.5 (88.89%) |
| | Viável | 6,2 (32,9%) | 2,2 (100%) | 9.2 (79.34%) |
| | Real | 7 (32,4%) | 2,6 (100%) | 9.9(86.5%) |

## 6.3 AMEAÇAS À VALIDADE

Nesta seção são apresentadas algumas ameaças que podem invalidar os resultados obtidos. Elas são descritas a seguir.

- Foi utilizada a ferramenta FMTS, que faz uso do framework FaMa para lidar com restrições do FM e derivar produtos. Este framework possui algumas limitações relacionadas à escalabilidade. No entanto, a abordagem é independente da ferramenta que está sendo utilizada para derivar os produtos do FM.

- Os FMs utilizados no experimento são pequenos quando comparados aos FM usados na indústria. Mas os resultados fornecem algumas indicações sobre o uso bem-sucedido da abordagem, podendo encontrar conjuntos reduzidos de produtos, com boas coberturas de critérios de teste e próximas ao ponto de referência informado pelo testador.

- Os RPs foram selecionados considerando nosso conhecimento prévio sobre a fronteira de Pareto gerada pelo algoritmo NSGA-II reportada na literatura. Os resultados encontrados podem ser diferentes para outros RPs e ainda quando considerada a presença de um testador real.

- Os algoritmos de otimização são não determinísticos por isso foram adotados procedimentos e avaliações citadas na literatura para avaliar estes algoritmos. Eles foram executados 30 vezes e foi realizado o *tuning* de alguns parâmetros específicos. Os demais parâmetros assumiram valores estabelecidos em trabalhos da literatura. A utilização de outros parâmetros pode resultar em resultados diferentes.

## 6.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados da avaliação experimental conduzida com dois PEMOAs, seis algoritmos baseados em HH e dois algoritmos tradicionais da literatura. Foram criados dois experimentos distintos: i) com 2 objetivos: número de produtos e escore de mutação e ii) com 3 objetivos: número de produtos, escore de mutação e cobertura *pairwise*. Esses experimentos tiveram seus parâmetros ajustados para as quatro LPS e cada algoritmo utilizado, que posteriormente foram executados 30 vezes com as melhores configurações encontradas.

Os resultados foram comparados com base nos indicadores Hipervolume com R-Metric (R-HV), distância euclidiana e número e porcentagem de soluções na ROI. Além disso, para cada experimento foram utilizados três pontos de referência estruturados usando conhecimento prévio sobre a fronteira de Pareto resultante do algoritmo NSGA-II, sendo eles um ponto inviável, viável e real. Após as execuções realizou-se uma análise quantitativa dos resultados obtidos.

De maneira geral, os resultados obtidos validam a abordagem proposta. Os PEMOAs são capazes de gerar um conjunto reduzido de soluções que melhor atendem às necessidades do DM, o mesmo foi observado pelas HHs baseadas em preferência. Além disso, a abordagem proposta apresentou resultados melhores em relação ao algoritmo NSGA-II e NSGA-II-HH, considerados até o momento os melhores para obter soluções para o problema descrito.

Em resumo a resposta da questão de pesquisa 1 (QP-1), que avalia os PEMOAs em relação ao algoritmo NSGA-II, observa-se que os PEMOAs, r-NSGA-II e R-NSGA-II, geram resultados melhores do que o algoritmo NSGA-II, em todos os casos. Além disso, o algoritmo R-NSGA-II apresenta os melhores resultados no geral, sendo considerado uma melhor opção para o testador por produzir um maior número de soluções na ROI. Já o algoritmo r-NSGA-II torna-se uma melhor opção no caso do testador desejar uma única solução na ROI.

A QP-2 avaliou os métodos de seleção para cada HH baseada em preferência. Os resultados obtidos mostraram-se muito semelhantes entre os métodos FRR-MAB e método aleatório. Entretanto, no escopo geral, o método FRR-MAB apresenta soluções melhores, sendo utilizado para responder as demais questões de pesquisa. O método CF não apresentou resultados muito satisfatórios em relação aos outros dois métodos.

Na QP-3 foi avaliado os PEMOAs e as HHs baseadas em preferência considerando o método de seleção FRR-MAB. No geral, as hiper-heurísticas apresentaram resultados superiores aos PEMOAs. Além disso, a hiper-heurística r-NSGA-II-HH conseguiu obter uma melhora

significativa em relação ao PEMOA r-NSGA-II, fato este que não ocorreu para o algoritmo R-NSGA-II. A utilização do conceito de r-dominância no algoritmo r-NSGA-II-HH pode ter possibilitado tal diferença, tendo em vista que no R-NSGA-II-HH foi utilizada a relação de dominância de Pareto tradicional. Além disso, dado o fato de o algoritmo R-NSGA-II já apresentar bons resultados, como mostrado na QP-1, melhorá-los pode ser uma tarefa mais difícil.

Por fim, a QP-4 comparou as HHs baseadas em preferência com a melhor HH tradicional aplicada ao problema, NSGA-II-HH. Foi possível observar que o algoritmo NSGA-II-HH gerou bons resultados, entretanto, muitas das soluções geradas ficaram fora da ROI, o que não ocorre com as HHs baseadas em preferência. Ainda assim, no caso geral, as HHs baseadas em preferência obtêm resultados superiores em relação a HH tradicional.

# 7 CONCLUSÕES

Este trabalho introduziu uma abordagem baseada em preferência do usuário para derivar produtos para o teste de linha de produto de software. A abordagem é multi-objetivo e permite o uso de diferentes fatores que afetam o problema, levando em consideração as preferências e necessidades do testador. Como outra vantagem, a abordagem é a priori e com base no método de ponto de referência. Isso a torna diferente de outras abordagens encontradas no campo de SBSE, que são mono-objetivas e interativas, exigindo a intervenção do testador muitas vezes durante o processo de otimização, o que pode causar o problema da fadiga.

Observa-se que o método de RP é muito adequado ao problema de teste de LPS, porque muitas vezes um nível de cobertura ou o número de produtos desejados, pode ser facilmente estabelecido pelo testador. Sabendo disso, a ideia é reduzir o esforço, gerando um número menor de soluções não interessantes do ponto de vista do testador, facilitando a tarefa de seleção de uma solução no final do processo.

Para implementar a abordagem proposta utilizaram-se os PEMOAS R-NSGA-II e r-NSGA-HH, fazendo uso do framework jMetal. Estes algoritmos são baseados no algoritmo NSGA-II e fazem uso do método de RP para expressar os interesses do usuário. Ainda, foram implementados algoritmos hiper-heurísticos baseados em preferência considerando ambos os PEMOAs e três métodos de seleção, FRR-MAB, CF, e um método aleatório. Tais algoritmos permitem realizar a seleção automática de operadores genéticos, sendo definidas 12 LLHs, que consistem na combinação de operadores de mutação e cruzamento. Para a resolução do problema definiram-se três funções objetivos que visam a minimizar o número de produtos no conjunto de teste, e maximizar a quantidade de mutantes mortos pela abordagem e a quantidade de pares cobertos.

Os resultados obtidos validam a abordagem proposta. Observou-se que os PEMOAs, apresentam melhores resultados em relação ao melhor algoritmo encontrado na literatura NSGA-II. Além disso, são opções mais viáveis para o DM devido a produzirem uma fronteira de soluções não-dominadas centradas na região de interesse. Contudo, o algoritmo r-NSGA-II mostra-se uma melhor opção quando o testador está interessado em uma ou poucas soluções. Quando o testador busca um conjunto de soluções próximas ao RP informado, o algoritmo R-NSGA-II mostra-se como uma melhor opção.

Não foram observadas grandes diferenças ao usar diferentes conjuntos de objetivos e RPs. Parece que o desempenho do algoritmo r-NSGA-II aumenta com o número de objetivos, e se RPs próximos a $PF_{true}$ são fornecidos, mas isso deve ser explorado em experimentos futuros.

Com relação aos métodos de seleção de LLHs avaliados, pode-se observar uma prevalência de melhores resultados, no caso geral, por parte do método FRR-MAB. Contundo, o método aleatório por muitas vezes apresenta resultados muito próximos e até melhores do que o FRR-MAB. Já o método CF, de maneira geral, não apresenta grandes resultados em relação aos demais em ambos os algoritmos, r-NSGA-II-HH e R-NSGA-II-HH, respectivamente.

Considerando os algoritmos de HHs baseados em preferência, foi observado que há uma grande melhora do algoritmo r-NSGA-II-HH em relação ao algoritmo r-NSGA-II. O mesmo

ocorre para o algoritmo R-NSGA-II-HH em relação ao R-NSGA-II, porém essa melhora não é tão significativa como no outro caso. Porém, a utilização ainda é validada por facilitar a configuração e a escolha dos operadores de busca.

Quando comparados os algoritmos de HH baseados em preferência com o melhor algoritmo HH proposto na literatura para o mesmo problema, observa-se que a abordagem proposta produz melhores resultados, sendo melhor em quase todos os experimentos e em ambas as formulações. A HH tradicional gera uma quantidade de soluções muito grande, resultando em um grande número de soluções não interessantes para o testador, ou seja, soluções fora da ROI. Já as HHs baseadas em preferência geram as soluções mais concentradas na ROI, gerando poucas ou nenhuma solução fora dela.

Em resumo, pode-se concluir que os PEMOAs apresentaram bons resultados quando aplicados ao problema de testes de LPS. A utilização da abordagem de HH baseada em preferência obteve resultados melhores em relação aos demais algoritmos no caso geral. Além disso, ainda possibilitam uma maior flexibilidade para o testador, realizando a configuração e seleção automática dos operadores de busca. Ainda, o uso do método de ponto de referência apresentou-se uma opção simples e intuitiva para expressar as preferências do testador, com relação ao teste de LPS.

## 7.1 CONTRIBUIÇÕES

A principal contribuição deste trabalho consiste em oferecer e avaliar uma abordagem multiobjetivo baseada em preferência aplicada ao teste de LPS. Tal abordagem pode ser aplicada em outros problemas similares de engenharia de software. Ao decorrer deste trabalho, alguns artigos foram produzidos, sendo eles:

- LIMA, JACKSON A. PRADO ; GUIZZO, GIOVANI ; VERGILIO, SILVIA R. ; SILVA, ALAN P. C. ; JAKUBOVSKI FILHO, HELSON L.; EHRENFRIED, HENRIQUE V. . Evaluating Different Strategies for Reduction of Mutation Testing Costs. In: the 1st Brazilian Symposium, 2016, Maringa. Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST, 2016. p. 1. Contribuição com a implementação de algoritmo genético aplicado ao problema do trabalho e análise dos resultados. Trabalho realizando durante o estudo e aprendizado sobre teste de software.

- JAKUBOVSKI FILHO, HELSON L.; LIMA, JACKSON A. PRADO ; VERGILIO, SILVIA R. . Automatic Generation of Search-Based Algorithms Applied to the Feature Testing of Software Product Lines. In: the 31st Brazilian Symposium, 2017, Fortaleza. Proceedings of the 31st Brazilian Symposium on Software Engineering - SBES'17, 2017. p. 114. Implementação da abordagem, análise dos resultados, participação na escrita do trabalho. Foi implementada uma HH off-line de geração aplicada ao problema de teste de LPS. Trabalho realizado durante o aprendizado sobre EMOAs e HH.

- JAKUBOVSKI FILHO, HELSON L.; FERREIRA, THIAGO, N.; VERGILIO, SILVIA R. Preference Based Multi-Objective Algorithms Applied to the Variability Testing of Software Product Lines. Journal of Systems and Software, 2018 (SUBMETIDO). Trabalho referente a dissertação. Participação com a implementação dos algoritmos, análise dos resultados e escrita do trabalho. Foram avaliados os PEMOAs em relação aos EMOAs tradicionais aplicados ao problema abordado nesta dissertação.

- JAKUBOVSKI FILHO, HELSON L.; FERREIRA, THIAGO, N.; VERGILIO, SILVIA R. Incorporating User Preferences in a Software Product Line Testing Hyper-Heuristic Approach. Proceedings of the IEEE Congress on Evolutionary Computation (CEC'18), 2018 (ACEITO PARA APRESENTAÇÃO). Trabalho referente a dissertação. Participação com a implementação dos algoritmos, análise dos resultados e escrita do trabalho. Foram avaliadas as HHs baseadas em preferência em relação aos PEMOAs e uma HH tradicional aplicadas ao problema abordado nesta dissertação.

## 7.2 TRABALHOS FUTUROS

A seguir são descritos alguns possíveis trabalhos futuros gerados a partir desse trabalho.

- Repetir o estudo experimental para avaliar a escalabilidade, utilizando LPSs maiores e outras LPSs utilizadas na indústria.
- Fazer uso de outros algoritmos baseados em preferência e até mesmo outras hiper-heurísticas, como por exemplo uma HH de geração automática de PEMOAs ou o algoritmo g-NSGA-II, entre outros.
- Considerar a aplicação da abordagem em outros problemas de engenharia de software.
- Considerar um número maior de objetivos, inserindo um espaço de busca maior, dificultando a processo da busca.
- Considerar o testador na indústria, utilizando pontos de referência utilizados na prática.
- Utilizar técnicas de aprendizado de máquina para aprender o comportamento do testador, fazendo uso de diferentes aplicações do testador em diferentes sistemas de teste, criando com isso um aprendizado sobre diferentes problemas em teste podendo gerar RPs automaticamente de acordo com atributos do sistema em teste.

# REFERÊNCIAS


[1] S. F. Adra, I. Griffin e P. J. Fleming. A comparative study of progressive preference articulation techniques for multiobjective optimisation. Em: *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization, EMO'07*, páginas 908–921, Springer, Matsushima, Japan, 2007.

[2] P. Auer, N. Cesa-Bianchi e P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] M. P. Basgalupp, R. C. Barros, T. S. da Silva e A. C. de Carvalho. Software effort prediction: a hyper-heuristic decision-tree based approach. Em: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, páginas 1109–1116, 2013.

[4] S. Bechikh, M. Kessentini, L. B. Said e K. Ghédira. Preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. *Advances in Computers*, 98:141–207, 2015.

[5] D. Benavides, S. Segura e A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615–636, 2010.

[6] D. Benavides, S. Segura, P. Trinidad e A. Ruiz-Cortés. Fama: Tooling a framework for the automated analysis of feature models. Em: *First International Workshop on Variability Modelling of Software intensive Systems*, páginas 129–1334, 2007.

[7] D. Benavides, S. Trujillo e P. Trinidad. On the modularization of feature models. Em: *First European Workshop on Model Transformation*, páginas 134, 2005.

[8] J. Branke, K. Deb, K. Miettinen e R. Słowinski. Multiobjective optimization - interactive and evolutionary approaches. *Lecture Notes in Computer Science, Springer-Verlag, New York*, 5252, 2008.

[9] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan e R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1476–9360, 2013.

[10] K. Chakhlevitch e P. Cowling. *Hyperheuristics: recent developments*, páginas 3–29. Adaptive and multilevel metaheuristics, 2008.

[11] P. C. Clements e L. M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[12] C. A. Coello, G. T. Pulido e E. M. Montes. Current and future research trends in evolutionary multiobjective optimization. Em: *Information Processing with Evolutionary Algorithms*

*Advanced, Information and Knowledge Processing*, páginas 213–231, Springer London, 2005.

[13] C. C. Coello. An updated survey of GA-based multiobjective optimization technique. *ACM Computing Surveys*, 32(2):109–143, junho 2000.

[14] C. C. Coello. Fundamentals of evolutionary multi-objective optimization. *Industrial Electronics Handbook. Intelligent Systems*, 2010.

[15] C. C. Coello, G. B. Lamont e D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[16] C. A. Coello Coello, G. B. Lamont e D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag New York, Inc., Secaucus, USA, 2nd edition, October 2006.

[17] D. M. Cohen, S. R. Dalal, M. L. Fredman e G. C. Patton. The AETG system: An approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, 23(7):437–444, 1997.

[18] M. B. Cohen, M. B. Dwyer e J. Shi. Coverage and adequacy in software product line testing. Em: *Proceedings of the ISSTA 2006 Workshop on Role of Software Architecture for Testing and Analysis, ROSATEA'06*, páginas 53–63, New York, NY, USA, 2006.

[19] P. I. Cowling, G. Kendall e E. Soubeiga. A Hyperheuristic approach to scheduling a sales summit. Em: *Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling*, páginas 176–190, 2001.

[20] K. Deb, A. Pratap, S. Agarwal e T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[21] K. Deb, J. Sundar, U. Bhaskara e S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, 2(3):27–286, 2006.

[22] M. E. Delamaro, J. C. Maldonado e M. Jino. *Introdução ao Teste de Software*. Elsevier, 2007.

[23] R. A. DeMillo, R. J. Lipton e F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *IEEE Computer*, 11(4):34–41, 1978.

[24] J. J. Durillo e A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[25] E. Engström e P. Runeson. Software product line testing - a systematic mapping study. *Information and Software Technology*, 53(1):2–13, 2011.

[26] F. Ensan, E. Bagheri e D. Gašević. Evolutionary search-based test generation for software product line feature models. Em: *Proceedings of the 24th international conference on Advanced Information Systems Engineering, CAiSE'12*, páginas 613–628, Springer-Verlag Berlin, 2012.

[27] R. Feldt. An interactive software development workbench based on biomimetic algorithms. Relatório técnico abs/1211.5451, Chalmers University of Technology, Gothenburg, Sweden, 2002.

[28] J. M. Ferreira. Teste de linha de produto de software baseado em mutação do diagrama de características. Dissertação de mestrado, Universidade Federal do Paraná, Curitiba - PR, 2012.

[29] J. M. Ferreira, S. R. Vergilio e M. A. Quináia. A mutation approach to feature testing of software product lines. Em: *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering, SEKE'13*, páginas 232–237, Boston, USA, 2013. Knowledge Systems Institute Graduate School.

[30] J. M. Ferreira, S. R. Vergilio e M. A. Quinaia. Software product line testing based on feature model mutation. *International Journal of Software Engineering and Knowledge Engineering*, 27(05):817–839, 2017.

[31] T. N. Ferreira, J. N. Kuk, A. T. R. Pozo e S. R. Vergilio. Product selection based on upper confidence bound MOEA/D-DRA for testing software product lines. Em: *Proceedings of the Congress on Evolutionary Computation, CEC '16*, páginas 4135–4142, Vancouver, Canada, 2016. IEEE.

[32] T. N. Ferreira, J. A. Prado Lima, A. Strickler, J. N. Kuk, S. R. Vergilio e A. Pozo. Hyper-heuristic based product selection for software product line testing. *IEEE Computational Intelligence Magazine*, 12(2):34–45, 2017.

[33] T. N. Ferreira, S. R. Vergilio e J. T. de Souza. Incorporating user preferences in search-based software engineering: A systematic mapping study. *Information and Software Technology*, 90:55–69, 2017.

[34] P. Garrido e M. C. Riff. Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, 16(6):795–834, 2010.

[35] G. Guizzo, G. M. Fritsche, S. R. Vergilio e A. T. R. Pozo. A hyper-heuristic for the multi-objective integration and test order problem. Em: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO'15*, páginas 1343–1350. ACM, 2015.

[36] M. Harman, E. Burke, J. Clark e X. Yao. Dynamic adaptive search based software engineering. Em: *Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM'12*, páginas 1949–3770, ACM, 2012.

[37] M. Harman, Y. Jia, J. Krinke, W. B. Langdon, J. Petke e Y. Zhang. Search based software engineering for software product line engineering: A survey and directions for future work. Em: *Proceedings of the 18th International Software Product Line Conference, SPLC'14*, volume 1, páginas 5–18, Florence, Italy, September 2014. ACM.

[38] M. Harman e B. F. Jones. Search-based software engineering. *Information and software Technology*, 43(14):833–839, 2001.

[39] M. Harman, S. A. Mansouri e Y. Zhang. Search-based software engineering: Trends, techniques and applications. *Computing Surveys*, 45(1):1–61, 2012.

[40] C. Henard, M. Papadakis e P. Heymans e Y. Le Traon G. Perrouin J. Klein. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test suites for large software product lines. Relatório técnico abs/1211.5451, Computing Research Repository - CoRR, Novembro 2012.

[41] C. Henard, M. Papadakis e Y. Le Traon. Mutation-based generation of software product line test configurations. Em: *International Symposium on Search Based Software Engineering, SSBSE'14*, páginas 92–106, Fortaleza, Brazil, October 2014. Springer.

[42] C. Henard, M. Papadakis, G. Perrouin, J. Klein e Y. le Traon. Assessing software product line testing via model-based mutation: An application to similarity testing. Em: *Proceedings of the 6th International Conference on Software Testing, Verification and Validation, ICSTW'13*, páginas 188–197, Luxembourg, Luxembourg, 2013. IEEE.

[43] IEEE Computer Society Press. *IEEE Standard Glossary of Software Engineering Terminology*, 2002. Standard 610.12-1990.

[44] SEI-Software Engineering Institute. A framework for software product line practice, version 4.2. `http://www.sei.cmu.edu/productlines/frame_report/PL.essential.act.htm`, 2007. Acessado em 10/03/2017.

[45] Y. Jia, M. Cohen, M. Harman e J. Petke. Learning combinatorial interaction test generation strategies using hyperheuristic search. Em: *Proceedings of the 37th International Conference on Software Engineering, ICSE'15*, volume 1, 2015.

[46] Li K, Kalyanmoy D. e Xin Y. R-metric: Evaluating the performance of preference-based evolutionary multi-objective optimization using reference points. *IEEE Transactions on Evolutionary Computation*, 2017.

[47] S. Kalboussi, S. Bechikh, M. Kessentini e L. Ben Said. Preference-based many-objective evolutionary testing generates harder test cases for autonomous agents. Em: *Proceedings of the 5th International Symposium on Search Based Software Engineering, SSBSE'13*, páginas 245–250, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[48] R. Kamalian, E. Yeh, Y. Zhang, A. M. Agogino A. M. e H. Takagi. Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems. Em: *IEEE International Conference on Fuzzy Systems*, páginas 678–684, 2006.

[49] A. Konak, D. W. Coit e A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering e System Safety*, 91(9):992–1007, 2006.

[50] W. H. Kruskal e W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[51] R. Kuhn, R. Kacker, Y. Lei e J. Hunter. Combinatorial software testing. *Computer*, 42(8):94–96, 2009.

[52] A. C. Kumari, K. Srinivas e M. P. Gupta. Software module clustering using a hyperheuristic based multi-objective genetic algorithm. Em: *Proceedings of the 3rd International Advance Computing Conference, IACC'13*, páginas 813–818, 2013.

[53] B. P. Lamancha e M. P. Usaola. Testing product generation in software product lines using pairwise for features coverage. Em: *Proceedings of the 22th International Conference on Testing Software and Systems, ICTSS'10*, páginas 111–125, Natal, Brazil, 2010. Springer.

[54] K. Li, A. Fialho, S. Kwong e Q. Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014.

[55] R. E. Lopez-Herrejon, L. Linsbauer e A. Egyed. A systematic mapping study of search-based software engineering for software product lines. *Information and Software Technology*, 61(C):33–51, 2015.

[56] M. Maashi, E. Özcan e G. Kendall. *A multi-objective hyper-heuristic based on choice function*, páginas 4475–4493. Expert Systems with Applications, 2014.

[57] B. Marculescu, R. Feldt e R. Torkar. A concept for an interactive search-based software testing system. Em: *Search Based Software Engineering, proceedings 4th International Symposium, SSBSE'12*, páginas 273–278, 2012.

[58] B. Marculescu, R. Feldt e R. Torkar. Objective re-weighting to guide an interactive search based software testing system. Em: *Proceedings of the 12th International Conference on Machine Learning and Applications, ICMLA'13*, 2013.

[59] B. Marculescu, R. Feldt, R. Torkar e S. Poulding. *An initial industrial evaluation of interactive search-based testing for embedded software*, volume 29, páginas 26–39. Applied Soft Computing, Elsevier, 2014.

[60] R. A. Matnei Filho e S. R. Vergilio. A mutation and multi-objective test data generation approach for feature testing of software product lines. Em: *Proceedings of the 29th Brazilian Symposium on Software Engineering, SBES'15*, páginas 21–30, Belo Horizonte , Brazil, September 2015. IEEE Computer Society.

[61] R. A. Matnei Filho e S. R. Vergilio. A multi-objective test data generation approach for mutation testing of feature models. *Journal of Software Engineering Research and Development*, 4(1), 2016.

[62] J. D. McGregor. Toward a fault model for software product lines. Em: *12th International Software Product Line Conference, SPLC'08*, páginas 157–162, 2008.

[63] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[64] M. W. Mkaouer, M. kessentini, S. Bechikh, K. Deb e M. Ó. Cinnéide. Recommendation system for software refactoring using innovization and interactive dynamic optimization. Em: *Proceedings of the 29th ACM international conference on Automated software engineering, ASE '14*, páginas 331–336, Vasteras, Sweden, 2014.

[65] G. J. Myers e C. Sandler. *The Art of Software Testing*. John Wiley e Sons, 2004.

[66] P. A. D. M. Silveira Neto, I. C. Machado, J. D. McGregor, E. S. Almeida e S. R. Lemos Meira. A systematic mapping study of software product lines testing. *Information and Software Technology*, 53(5):407–423, 2011.

[67] C. Nie e H. Leung. A survey of combinatorial testing. *ACM Computing Surveys (CSUR)*, 43(2):11:1–11:29, 2011.

[68] S. Oster, M. Zink, M. Lochau e M. Grechanik. Pairwise feature-interaction testing for SPLs: Potentials and limitations. Em: *Proceedings of the 15th International Software Product Line Conference, SPLC'11*, volume 2, páginas 1–8, Munich, Germany, 2011. ACM.

[69] V. Pareto. Manuel d'economie politique. *Ams Press*, 1927.

[70] G. Perrouin, S. Sen, J. Klein, B. Baudry e Y. le Traon. Automated and scalable T-wise test case generation strategies for Software Product Lines. Em: *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation, ICST'10*, páginas 459–468, 2010.

[71] K. Pohl, G. Böckle e F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[72] K. Pohl e A. Metzger. Software Product Line testing. *Communications of the ACM*, 49(12):78–81, 2006.

[73] R. S. Pressman. *Engenharia de Software*, páginas 787–822. Pearson, 2010.

[74] A. R. C. Rocha, C. Maldonado J e K. C. Weber. *Qualidade de Software*. Prentice Hall, 2001.

[75] S. Ali S. Wang e A. Gotlieb. Minimizing test suites in software product lines using weight-based genetic algorithms. Em: *Proceedings of the Annual Conference on Genetic and Evolutionary Computation, GECCO '13*, páginas 1493–1500, New York, NY, USA, 2013.

[76] N.R. Sabar, M. Ayob, G. Kendall e R. Qu. A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, páginas 217–228, 2015.

[77] L. B. Said, S. Bechikh e K. Ghédira. The r-dominance: A new dominance relation for interactive evolutionary multicriteria decision making. *IEEE Transactions on Evolutionary Computation*, 14(5):801–818, 2010.

[78] A. S. Sayyad, T. Menzies e H. Ammar. On the value of user preferences in searchbased software engineering: A case study in software product lines. Em: *Proceedings of the 2013 International Conference on Software Engineering, ICSE'13*, páginas 492–501, IEEE Press, 2013.

[79] S. Segura, R. M. Hierons, D. Benavides e A. Ruiz-Cortés. Automated test data generation on the analyses of feature models: A metamorphic testing approach. Em: *Proceedings of the 3th International Conference on Software Testing, Verification and Validation, ICST'09*, páginas 35–44, 2009.

[80] Z. Stephenson, Y. Zhan, J. Clark e J. McDermid. Test data generation for product lines - a mutation testing approach. Em: *Proceedings of the International Workshop on Software Product Line Testing, SPLIT'04*, páginas 231–237, 2004.

[81] A. Strickler, J. A. Prado Lima, S. R. Vergilio e A. T. R. Pozo. Deriving products for variability test of feature models with a hyper-heuristic approach. *Applied Soft Computing*, 49:1232–1242, 2016.

[82] K. Tai e Y. Lei. A test generation strategy for pairwise testing. *IEEE Transactions on Software Engineering*, 28(1):109–111, 2002.

[83] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[84] A. Tevanlinna, J. Taina e R. Kauppinen. Product family testing: a survey. *ACM SIGSOFT Software Engineering*, 29(2):12, 2004.

[85] P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura e A. Jimenez. FaMa framework. Em: *Proceedings of the 12th International Software Product Line Conference, SPLC'08*, páginas 359–359, Limerick, Ireland, 2008. IEEE.

[86] F. J. van der Linden, K. Schmid e E. Rommes. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag, 2007.

[87] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan e M. Liaaen. Multi-objective test prioritization in software product line testing: An industrial case study. Em: *Proceedings of the 18th International Software Product Line Conference, SPLC'14*, páginas 32–41, 2014.

[88] S. Weißleder, D. Sokenou e B. Schlingloff. Reusing state machines for automatic test generation in product lines. Em: *1st workshop on model-based testing in practice, MoTiP'08*, páginas 19, 2008.

[89] A. P. Wierzbicki. *The Use of Reference Objectives in Multiobjective Optimization*, páginas 468–486. Springer, 1980.

[90] A. E. E. Yamany e M. S. Elgamel. Smart Optiselect preference based innovative framework for user-in-the-loop feature selection in software product lines. Em: *Proceedings of the 7th International Conference on Information Technology, ICIT'15*, páginas 657–666, 2015.

[91] A. E. E. Yamany, M. Shaheen e A. S. Sayyad. OPTI-SELECT: An interactive tool for user-in-the-loop feature selection in software product lines. Em: *Proceedings of the 18th International Software Product Line, SPLC'14*, páginas 126–129, Florence, Italy, 2014.

[92] R. E. z Herrejon, J. F. Chicano, J. Ferrer, A. Egyed e E. Alba. Multi-objective optimal test suite computation for software product line pairwise testing. Em: *International Conference on Software Maintenance, ICSM'13*, páginas 404–407, IEEE Computer Society, 2013.

[93] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Tese de doutorado, ETH Zurich, Switzerland, 1999.

[94] E. Zitzler, M. Laumanns e L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. Em: *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, páginas 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering.

[95] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca e V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

# APÊNDICE A: MELHORES CONFIGURAÇÕES DE PARÂMETROS

Este apêndice apresenta as tabelas contendo os resultados das melhores configurações de parâmetros para cada formulação, FM e algoritmo utilizados nos experimentos descritos no Capítulo 6. A Tabela A.1 apresenta os melhores valores para a população e número de avaliações para os algoritmos r-NSGA-II e R-NSGA-II. A Tabela A.2 apresenta os dois parâmetros da mesma forma, porém para os algoritmo r-NSGA-II-HH-CF, r-NSGA-II-HH-FRR e r-NSGA-II-HH-R. Por fim, a Tabela A.2 apresenta os dois parâmetros da mesma forma, porém para os algoritmo R-NSGA-II-HH-CF, R-NSGA-II-HH-FRR e R-NSGA-II-HH-R.

Tabela A.1: Melhores configurações definidas para os PEMOAs.

| Formulação | FM | Algoritmo | População | Max. de Aval. |
|---|---|---|---|---|
| 2-Objetivos | James | r-NSGA-II | 50 | 30000 |
| | | R-NSGA-II | 200 | 30000 |
| | CAS | r-NSGA-II | 50 | 90000 |
| | | R-NSGA-II | 200 | 30000 |
| | WS | r-NSGA-II | 50 | 60000 |
| | | R-NSGA-II | 50 | 60000 |
| | E-Shop | r-NSGA-II | 200 | 90000 |
| | | R-NSGA-II | 200 | 60000 |
| 3-Objetivos | James | r-NSGA-II | 200 | 90000 |
| | | R-NSGA-II | 200 | 90000 |
| | CAS | r-NSGA-II | 200 | 30000 |
| | | R-NSGA-II | 50 | 60000 |
| | WS | r-NSGA-II | 50 | 60000 |
| | | R-NSGA-II | 200 | 60000 |
| | E-Shop | r-NSGA-II | 100 | 60000 |
| | | R-NSGA-II | 200 | 90000 |

Tabela A.2: Melhores configurações definidas para r-NSGA-HH.

| Formulação | FM | Algoritmo | População | Max. de Aval. |
|---|---|---|---|---|
| 2-Objetivos | James | r-NSGA-II-CF | 50 | 90000 |
| | | r-NSGA-II-FRR | 50 | 60000 |
| | | r-NSGA-II-R | 50 | 90000 |
| | CAS | r-NSGA-II-CF | 200 | 30000 |
| | | r-NSGA-II-FRR | 100 | 90000 |
| | | r-NSGA-II-R | 50 | 90000 |
| | WS | r-NSGA-II-CF | 100 | 90000 |
| | | r-NSGA-II-FRR | 50 | 90000 |
| | | r-NSGA-II-R | 100 | 90000 |
| | E-Shop | r-NSGA-II-CF | 100 | 60000 |
| | | r-NSGA-II-FRR | 100 | 90000 |
| | | r-NSGA-II-R | 50 | 60000 |
| 3-Objetivos | James | r-NSGA-II-CF | 50 | 90000 |
| | | r-NSGA-II-FRR | 50 | 60000 |
| | | r-NSGA-II-R | 100 | 90000 |
| | CAS | r-NSGA-II-CF | 50 | 60000 |
| | | r-NSGA-II-FRR | 200 | 30000 |
| | | r-NSGA-II-R | 200 | 30000 |
| | WS | r-NSGA-II-CF | 100 | 90000 |
| | | r-NSGA-II-FRR | 200 | 90000 |
| | | r-NSGA-II-R | 50 | 90000 |
| | E-Shop | r-NSGA-II-CF | 50 | 30000 |
| | | r-NSGA-II-FRR | 100 | 60000 |
| | | r-NSGA-II-R | 200 | 60000 |

Tabela A.3: Melhores configurações definidas para R-NSGA-HH.

| Formulação | FM | Algoritmo | População | Max. de Aval. |
|---|---|---|---|---|
| 2-Objetivos | James | R-NSGA-II-CF | 50 | 30000 |
| | | R-NSGA-II-FRR | 50 | 60000 |
| | | R-NSGA-II-R | 50 | 90000 |
| | CAS | R-NSGA-II-CF | 50 | 90000 |
| | | R-NSGA-II-FRR | 50 | 30000 |
| | | R-NSGA-II-R | 50 | 30000 |
| | WS | R-NSGA-II-CF | 100 | 30000 |
| | | R-NSGA-II-FRR | 50 | 90000 |
| | | R-NSGA-II-R | 50 | 90000 |
| | E-Shop | R-NSGA-II-CF | 50 | 30000 |
| | | R-NSGA-II-FRR | 100 | 30000 |
| | | R-NSGA-II-R | 50 | 60000 |
| 3-Objetivos | James | R-NSGA-II-CF | 50 | 90000 |
| | | R-NSGA-II-FRR | 50 | 60000 |
| | | R-NSGA-II-R | 100 | 90000 |
| | CAS | R-NSGA-II-CF | 100 | 90000 |
| | | R-NSGA-II-FRR | 50 | 30000 |
| | | R-NSGA-II-R | 50 | 90000 |
| | WS | R-NSGA-II-CF | 50 | 60000 |
| | | R-NSGA-II-FRR | 50 | 60000 |
| | | R-NSGA-II-R | 50 | 90000 |
| | E-Shop | R-NSGA-II-CF | 200 | 90000 |
| | | R-NSGA-II-FRR | 50 | 30000 |
| | | R-NSGA-II-R | 200 | 30000 |

# APÊNDICE B: QP-2: FRR-MAB X CF X ALEATÓRIO - RESULTADOS

Neste apêndice são apresentados os resultados obtidos para responder à questão de pesquisa 2 (QP-2).

**Experimento com 2 objetivos**

A Tabela B.1, apresenta os resultados obtidos utilizando o algoritmo r-NSGA-II-HH com os métodos de seleção e o indicador de qualidade R-HV. De maneira geral o algoritmo r-NSGA-II-HH-FRR apresenta resultados melhores em relação aos demais algoritmos. Entretanto, o algoritmo r-NSGA-II-HH-R é melhor ou estatisticamente equivalente ao melhor resultado em todos os FMs e RPs. Já o algoritmo r-NSGA-II-HH-CF apresenta resultados inferiores aos demais, tendo em vista que nos dois únicos casos onde obteve resultado superior, estes foram estatisticamente equivalentes.

Tabela B.1: R-HV para 2 objetivos para o algoritmo r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | 0.855482 (0.009322) | 0.849487 (0.011108) | **0.859989 (0.010514)** |
| | Viável | 0.916964 (0.004584) | 0.919008 (0.004812) | **0.917532 (0.004599)** |
| | PFTrue | 0.897359 (0.005228) | **0.898279 (0.004208)** | 0.897819 (0.004768) |
| CAS | Inviável | **0.947791 (0.008699)** | 0.937531 (0.024251) | 0.944709 (0.009631) |
| | Viável | 0.947893 (0.00885) | **0.954228 (0.008669)** | 0.953241 (0.007969) |
| | PFTrue | 0.94635 (0.008185) | 0.952888 (0.005434) | **0.954116 (0.003543)** |
| WS | Inviável | 0.948839 (0.015598) | **0.953677 (0.005914)** | 0.94946 (0.021933) |
| | Viável | 0.956743 (0.009871) | 0.953761 (0.007617) | **0.959862 (0.008921)** |
| | PFTrue | **0.962356 (0.00188)** | 0.962274 (8.87E-4) | 0.962138 (0.001499) |
| E-Shop | Inviável | 0.957856 (0.00741) | **0.961268 (0.009387)** | 0.957126 (0.007336) |
| | Viável | 0.954936 (0.008342) | **0.955645 (0.007969)** | 0.952471 (0.006834) |
| | PFTrue | 0.974658 (0.003055) | **0.977753 (0.003957)** | 0.974985 (0.002997) |

A Tabela B.2 mostra os resultados de R-HV considerando os métodos de seleção em combinação com o algoritmo R-NSGA-II-HH. É possível observar que os algoritmos R-NSGA-II-HH-FRR e R-NSGA-II-HH-R são estatisticamente equivalentes em todos os FMs e RPs. É importante notar que para as instâncias que mais derivam produtos, WS e E-Shop, o algoritmo R-NSGA-II-HH-FRR apresenta valores levemente superiores, enquanto que o algoritmo R-NSGA-II-HH-R apresenta para todos os casos melhores valores para RPs reais. Ainda, pode-se notar que o algoritmo R-NSGA-II-HH-CF não obteve melhor resultado e nem equivalência estatística ao melhor valor em nenhum caso.

Tabela B.2: R-HV para 2 objetivos para o algoritmo R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | 0.597358 (0.127455) | 0.666078 (0.110133) | **0.672859 (0.112391)** |
| | Viável | 0.737774 (0.200743) | **0.882451 (0.00691)** | 0.881463 (0.002756) |
| | PFTrue | 0.788655 (0.203722) | 0.926268 (0.009816) | **0.931736 (0.009429)** |
| CAS | Inviável | 0.628942 (0.231708) | 0.900968 (0.07969) | **0.921526 (0.05756)** |
| | Viável | 0.724192 (0.28965) | 0.97387 (0.009275) | **0.974878 (0.008944)** |
| | PFTrue | 0.753246 (0.290403) | **0.975569 (0.004943)** | 0.975453 (0.008788) |
| WS | Inviável | 0.661481 (0.254544) | **0.929605 (0.051043)** | 0.900005 (0.088998) |
| | Viável | 0.751659 (0.284532) | **0.967229 (0.009898)** | 0.962394 (0.01015) |
| | PFTrue | 0.730963 (0.288177) | 0.979089 (0.002602) | **0.979193 (0.002139)** |
| E-Shop | Inviável | 0.640376 (0.286317) | **0.951833 (0.058212)** | 0.949725 (0.068351) |
| | Viável | 0.630091 (0.317007) | **0.966083 (0.012696)** | 0.960863 (0.013619) |
| | PFTrue | 0.761815 (0.305497) | 0.9869 (0.004791) | **0.988589 (0.003268)** |

Considerando o indicador DE, e a Tabela B.3 que apresenta os resultados para o algoritmo r-NSGA-II-HH com os 3 métodos de seleção é possível notar que para a menor instância (James) todos os algoritmos obtiveram a mesma solução ou seja, foram equivalentes. Entretanto, para os demais casos, fica difícil concluir qual algoritmo foi melhor. Porém, em um escopo geral, o algoritmo r-NSGA-II-HH-FRR obtém um resultado a mais em relação aos outros algoritmos, obtendo às melhores soluções em 8 de 12 casos e os outros dois algoritmos empatam no número de casos (7 de 12).

Tabela B.3: DE para o experimento com 2 objetivos com o algoritmo r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **0.062787 (4; 92.4)** | **0.062787 (4; 92.4)** | **0.062787 (4; 92.4)** |
| | Viável | **0.034431 (5; 98.1)** | **0.034431 (5; 98.1)** | **0.034431 (5; 98.1)** |
| | PFTrue | **0.011132 (5; 98.1)** | **0.011132 (5; 98.1)** | **0.011132 (5; 98.1)** |
| CAS | Inviável | **0.011941 (7; 97.7)** | 0.075857 (5; 89.4) | 0.015554 (6; 95.5) |
| | Viável | 0.026784 (8; 98.6) | **0.022489 (7; 98.2)** | 0.035595 (8; 99.5) |
| | PFTrue | 0.012577 (8; 98.2) | 0.016931 (8; 98.6) | **0.012379 (7; 98.2)** |
| WS | Inviável | 0.043558 (7; 92.7) | **0.010558 (8; 96.6)** | 0.051844 (7; 91.8) |
| | Viável | **0.020775 (9; 98.0)** | **0.020775 (9; 98.0)** | **0.020775 (9; 98.0)** |
| | PFTrue | **0.008796 (10; 98.8)** | **0.008796 (10; 98.8)** | **0.008796 (10; 98.8)** |
| E-Shop | Inviável | **0.004048 (7; 97.2)** | 0.020924 (6; 94.9) | 0.018408 (6; 95.1) |
| | Viável | 0.034869 (8; 98.4) | 0.024863 (7; 97.4) | **0.007682 (7; 95.6)** |
| | PFTrue | 0.012386 (9; 99.2) | **0.007361 (8; 98.7)** | 0.012386 (9; 99.2) |

Novamente, a Tabela B.4 apresenta a solução mais próxima dos RPs geradas pelos algoritmos levando em consideração o conjunto $PF_{known}$. Dessa forma, são apresentados os valores de DE entre a solução e os RPs, e a solução selecionada (a mais próxima) considerando os métodos de seleção com o algoritmo R-NSGA-II. Os valores em negrito representam os melhores resultados.

É possível notar na tabela que o algoritmo R-NSGA-II-HH-FRR apresenta uma prevalência de melhores resultados para todos os FMs e RPs. No FM James nos pontos inviáveis e viáveis os algoritmos apresentam resultados equivalentes. É importante notar que o algoritmo R-NSGA-II-HH-FRR apresenta melhores resultados nos pontos viáveis e reais em todos os

Tabela B.4: DE para o experimento com 2 objetivos com o algoritmo R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **0.197529 (3; 78.3)** | **0.197529 (3; 78.3)** | **0.197529 (3; 78.3)** |
| | Viável | **0.050000 (6; 100.0)** | **0.050000 (6; 100.0)** | **0.050000 (6; 100.0)** |
| | Real | 0.033411 (6; 100.0) | **0.011132 (5; 98.1)** | 0.033411 (6; 100.0) |
| CAS | Inviável | 0.110991 (4; 85.9) | 0.119800 (4; 85.0) | **0.115396 (4; 85.4)** |
| | Viável | 0.040062 (9; 100.0) | **0.009429 (7; 96.9)** | 0.026784 (8; 98.6) |
| | Real | 0.030327 (9; 100.0) | **0.000837 (7; 96.9)** | 0.030327 (9; 100.0) |
| WS | Inviável | 0.135324 (5; 83.4) | **0.046315 (7; 92.4)** | 0.138124 (5; 83.1) |
| | Viável | 0.040049 (12; 100.0) | **0.018033 (9; 97.7)** | 0.034398 (11; 99.4) |
| | Real | 0.013757 (9; 96.6) | **0.008796 (10; 98.8)** | 0.014534 (11; 99.4) |
| E-Shop | Inviável | **0.076619 (5; 89.3)** | 0.094381 (5; 87.5) | 0.127363 (4; 84.2) |
| | Viável | 0.047462 (11; 99.7) | **0.042395 (10; 99.2)** | 0.047462 (11; 99.7) |
| | Real | 0.020169 (12; 100.0) | **0.014949 (10; 99.4)** | 0.017548 (11; 99.7) |

FMs. Além disso, apresenta resultado inferior em relação aos outros algoritmos no ponto inviável nas instâncias CAS e E-Shop, onde respectivamente os algoritmos R-NSGA-II-HH-R e R-NSGA-II-HH-CF, foram melhores.

A Tabela B.5 mostra o número médio de soluções na ROI e a taxa entre o número de soluções encontradas na ROI e o número total de soluções encontradas, pelo algoritmo r-NSGA-II-HH em conjunto com os 3 métodos de seleção. Os valores em negrito representam o menor número de soluções na ROI. Já a Tabela B.6 apresenta respectivamente os mesmos resultados, porém com os valores encontrados pelo algoritmo R-NSGA-II-HH com os métodos de seleção.

Tabela B.5: Número e porcentagem de soluções para o experimento de 2 objetivos com r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **1.0 (100%)** | **1.0 (100%)** | **1.0 (100%))** |
| | Viável | **2.0 (100%)** | **2.0 (100%)** | **2.0 (100%)** |
| | Real | **1.0 (100%)** | **1.0 (100%)** | **1.0 (100%)** |
| CAS | Inviável | **1.6 (100%)** | 2.0 (97.77%) | **1.6 (100%)** |
| | Viável | **1.3 (100%)** | 1.6 (100%) | **1.3 (100%)** |
| | Real | 1.5 (100%) | **1.1 (100%)** | **1.1 (100%)** |
| WS | Inviável | 2.0 (97.77%) | **1.8 (100%)** | 2.0 (96.66%) |
| | Viável | 2.2 (100%) | **2.1 (100%)** | 2.5 (100%) |
| | Real | 1.6 (100%) | 1.7 (100%) | **1.5 (100%)** |
| E-Shop | Inviável | 1.8 (100%) | 2.1 (100%) | **1.4 (100%)** |
| | Viável | 2.4 (100%) | 2.5 (100%) | **2.1 (100%)** |
| | Real | **1.7 (100%)** | 2.2 (100%) | **1.7 (100%)** |
| Média | Inviável | 1.6 (99.44%) | 1.7 (99.44%) | 1.3 (99.16%) |
| | Viável | 1.9 (100%) | 2.1 (100%) | 1.9 (100%) |
| | Real | 1.4 (100%) | 1.5 (100%) | 1.3 (100%) |

De maneira geral, observa-se na Tabela B.5 que o comportamento dos algoritmos foram bem semelhantes, ocorrendo uma varição pequena em relação à média. O que se pode notar é que para o RP inviável os algoritmos não obtêm em média 100% das soluções na ROI, embora

os valores obtidos estejam bem próximos disso. Mais especificamente, é possível notar que o algoritmo r-NSGA-II-HH-R gera o menor número médio de soluções, que na grande maioria dos casos estão 100% na ROI. Isso não ocorre em apenas um caso, no qual são usados RPs inviáveis. Mas neste caso, em média 99.16% das soluções estão na ROI. Já o algoritmo r-NSGA-II-HH-FRR pode gerar os melhores *trade-offs*, ou seja, ele gera mais soluções e a maioria delas está dentro da ROI.

Tabela B.6: Número e porcentagem de soluções para o experimento de 2 objetivos com R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **1.4 (48.38%)** | 1.4 (19.98%) | 1.4 (19.41%) |
| | Viável | **2.5 (58.11%)** | 3.0 (33.78%) | 3.0 (32.48%) |
| | Real | **2.6 (54.29%)** | 3.2 (41.71%) | 3.0 (44.44%) |
| CAS | Inviável | **2.7 (60.75%)** | 5.0 (52.47%) | 5.4 (57.41%) |
| | Viável | **3.4 (73.09%)** | 5.1 (85.67%) | 5.2 (84.95%) |
| | Real | **3.1 (80.84%)** | 5.6 (80.87%) | 5.5 (80.10%) |
| WS | Inviável | **3.3 (65.31%)** | 6.1 (59.62%) | 5.7 (53.87%) |
| | Viável | **4.1 (81.60%)** | 5.3 (89.75%) | 5.8 (80.89%) |
| | Real | **3.7 (86.62%)** | 6.0 (76.71%) | 6.2 (70.67%) |
| E-Shop | Inviável | **2.9 (78.74%)** | 6.6 (64.73%) | 7.1 (60.44%) |
| | Viável | **3.3 (86.07%)** | 6.9 (83.16%) | 6.8 (82.41%) |
| | Real | **3.7 (87.45%)** | 6.9 (85.10%) | 7.3 (77.80%) |
| Média | Inviável | 2.5 (63.30%) | 4.7 (49.2%) | 4.9 (47.8%) |
| | Viável | 3.3 (74.71%) | 5.1 (73.1%) | 5.2 (71.18%) |
| | Real | 3.2 (77.3%) | 5.4 (71.1%) | 5.5 (68.25%) |

Na Tabela B.6 é possível observar que o algoritmo R-NSGA-II-HH-CF gera o menor número de soluções em todos os FMs e RPs, além disso, apresenta para a grande maioria dos casos a maior porcentagem de soluções dentro da ROI, variando de 63.30% a 74.71%. Os algoritmos R-NSGA-II-HH-FRR e R-NSGA-II-HH-R produzem um maior número de soluções na ROI, sendo que em média, a diferença entre os dois RPs é muito pequena em todos os casos, porém, prevalece um número levemente maior para o algoritmo R-NSGA-II-HH-R. Entretanto, o algoritmo R-NSGA-II-HH-FRR obtém porcentagens de soluções na ROI superiores, variando de 49.2% a 73.1% em relação a 47.8% a 71.18% obtidos pelo R-NSGA-II-HH-R.

Resumindo, os resultados para a formulação com 2 objetivos, o algoritmo r-NSGA-II-HH-R apresenta melhor resultado ou estatisticamente equivalente ao melhor em todos os casos quando o indicador R-HV é utilizado. Já o algoritmo r-NSGA-II-HH-FRR apresenta resultados mais próximos ao RP, sendo melhor aplicável quando o testador tem preferência por um conjunto maior de boas soluções próximas a sua região de interesse. Em caso oposto, o algoritmo r-NSGA-II-HH-R apresenta-se como uma melhor opção.

De maneira geral ocorre uma equivalência entre os algoritmos R-NSGA-II-HH-FRR e R-NSGA-II-HH-R com relação a diversidade e convergência em todos os FMs e RPs. Este fato é demostrado pelo indicador R-HV e também pela porcentagem e número médio de soluções na ROI. Porém, o algoritmo R-NSGA-II-HH-FRR se destaca no indicador DE, apresentando a solução mais próxima em relação ao RP em quase todos os FMs e RPs.

O algoritmo R-NSGA-II-HH-CF destaca-se apenas no número de soluções na ROI, mostrando-se como uma melhor opção quando o testador deseja um *trade-off* reduzido de

soluções na ROI. Além disso, apresentou as melhores médias de porcentagem de soluções na ROI em todos os RPs.

### Experimento com 3 objetivos

Nos resultados apresentados na Tabela B.7, são mostradas as médias de R-HV para o algoritmo r-NSGA-II-HH com os 3 métodos de seleção. Pode-se notar que os três algoritmos possuem resultados equivalentes em quase todos os FMs e RPs, exceto no FM CAS e RP inviável, no qual o algoritmo r-NSGA-II-HH-CF não é equivalente ao melhor resultado. Embora o algoritmo r-NSGA-II-HH-FRR apresente resultados levemente superiores em relação ao algoritmo r-NSGA-II-HH-R, não se pode concluir que este seja melhor devido a sua equivalência estatística.

Tabela B.7: R-HV para 3 objetivos para o algoritmo r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **0.792386 (0.008579)** | 0.789351 (0.011961) | 0.788778 (0.010511) |
| | Viável | 0.872334 (0.074826) | 0.861724 (0.080063) | **0.898209 (0.046101)** |
| | PFTrue | **0.799615 (0.033872)** | 0.788002 (0.031475) | 0.796353 (0.022775) |
| CAS | Inviável | 0.887191 (0.02812) | **0.91302 (0.018707)** | 0.903699 (0.025871) |
| | Viável | 0.865116 (0.038786) | **0.884443 (0.039629)** | 0.866851 (0.043211) |
| | PFTrue | 0.917761 (0.025074) | 0.92782 (0.026705) | **0.93233 (0.014714)** |
| WS | Inviável | 0.914225 (0.025376) | **0.919196 (0.021416)** | 0.91727 (0.020971) |
| | Viável | **0.967759 (0.00994)** | 0.95642 (0.009214) | 0.960404 (0.010194) |
| | PFTrue | 0.941483 (0.007686) | **0.943381 (0.008104)** | 0.943068 (0.008304) |
| E-Shop | Inviável | 0.96102 (0.007337) | 0.956346 (0.014745) | **0.963578 (0.009403)** |
| | Viável | 0.956328 (0.01208) | **0.961349 (0.011311)** | 0.958666 (0.016088) |
| | PFTrue | 0.925029 (0.017077) | 0.923502 (0.025139) | **0.92613 (0.020129)** |

Da mesma forma, a Tabela B.8 apresenta as médias de R-HV para o algoritmo R-NSGA-II-HH com os métodos de seleção. É observado que o algoritmo R-NSGA-II-HH-R apresenta os melhores resultados na grande maioria dos casos, entretanto, sem diferença estatística com o algoritmo R-NSGA-II-HH-FRR, em todos os casos onde foi melhor. Além disso, para o FM E-Shop e RP real, o algoritmo R-NSGA-II-HH-FRR apresenta o melhor resultado, com diferença estatística para ambos os algoritmos. O algoritmo R-NSGA-II-HH-CF não foi melhor e nem estatisticamente equivalente ao melhor em nenhum caso.

A Tabela B.9 apresenta a solução mais próxima ao RP informado, por meio do indicador DE, para o algoritmo r-NSGA-II-HH em conjunto com os métodos de seleção. De maneira geral, o algoritmo r-NSGA-II-HH-R tem uma predominância de melhores resultados (8 de 12). No FM James e em todos os RPs, todos os algoritmos apresentaram os mesmos resultados, ou seja, foram equivalentes. Além disso, para a maioria dos casos o algoritmo r-NSGA-II-HH-R apresenta soluções melhores para os pontos viáveis e reais. Além disso, o algoritmo r-NSGA-II-HH-CF apresenta um número maior de melhores resultados (7 de 12) em relação ao algoritmo r-NSGA-II-HH-FRR (4 de 12).

A Tabela B.10 apresenta os resultados do indicador DE para o algoritmo R-NSGA-II-HH com os 3 métodos de seleção. Observa-se que todos os algoritmos possuem o mesmo número de melhores soluções (5 de 12). Devido a essa similaridade e distribuição das melhores soluções, torna-se difícil a definição do(s) melhor(es) método(s) para este caso.

Tabela B.8: R-HV para 3 objetivos para o algoritmo R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | 0.505104 (0.184117) | 0.690575 (0.110315) | **0.696519 (0.09645)** |
| | Viável | 0.739978 (0.234189) | 0.909189 (0.007094) | **0.909413 (0.006735)** |
| | PFTrue | 0.684406 (0.227977) | 0.834227 (0.015283) | **0.841259 (0.017393)** |
| CAS | Inviável | 0.579348 (0.29419) | 0.853895 (0.049637) | **0.87045 (0.046682)** |
| | Viável | 0.678654 (0.340623) | **0.900703 (0.063162)** | 0.892917 (0.048797) |
| | PFTrue | 0.656006 (0.321125) | 0.908511 (0.038665) | **0.913182 (0.035671)** |
| WS | Inviável | 0.584995 (0.317202) | **0.940178 (0.04266)** | 0.935823 (0.033406) |
| | Viável | 0.608764 (0.36546) | 0.961598 (0.011756) | **0.963958 (0.010366)** |
| | PFTrue | 0.652839 (0.35731) | 0.969412 (0.006575) | **0.970289 (0.007286)** |
| E-Shop | Inviável | 0.669008 (0.330951) | 0.949669 (0.038142) | **0.960486 (0.013733)** |
| | Viável | 0.663642 (0.370761) | **0.966667 (0.011692)** | 0.956843 (0.018439) |
| | PFTrue | 0.651059 (0.368715) | **0.958038 (0.024681)** | 0.944679 (0.022095) |

Tabela B.9: DE para o experimento com 3 objetivos com o algoritmo r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **0.073644 (4; 92.4; 100.0)** | **0.073644 (4; 92.4; 100.0)** | **0.073644 (4; 92.4; 100.0)** |
| | Viável | **0.063140 (4; 92.4; 98.6)** | **0.063140 (4; 92.4; 98.6)** | **0.063140 (4; 92.4; 98.6)** |
| | PFTrue | **0.060773 (4; 92.4; 100.0)** | **0.060773 (4; 92.4; 100.0)** | **0.060773 (4; 92.4; 100.0)** |
| CAS | Inviável | **0.028884 (6; 95.5; 99.4)** | 0.038932 (6; 94.2; 98.9) | 0.042245 (6; 93.8; 97.8) |
| | Viável | 0.037120 (6; 94.7; 99.4) | 0.035799 (7; 96.9; 99.4) | **0.020446 (7; 96.9; 97.8)** |
| | PFTrue | 0.028903 (6; 95.1; 99.4) | 0.022490 (7; 96.0; 100.0) | **0.012560 (7; 96.9; 99.4)** |
| WS | Inviável | 0.061164 (7; 91.0; 95.8) | **0.014769 (8; 96.0; 96.4)** | 0.058183 (7; 91.8; 94.3) |
| | Viável | 0.012109 (10; 98.5; 98.9) | 0.020794 (11; 99.4; 99.4) | **0.010696 (10; 98.8; 98.4)** |
| | PFTrue | **0.010966 (9; 97.7; 97.9)** | 0.015949 (9; 97.7; 97.4) | 0.01096 (9; 97.7; 97.9) |
| E-Shop | Inviável | **0.005211 (9; 98.9; 99.0)** | 0.025768 (7; 96.4; 99.0) | 0.028850 (8; 96.1; 99.5) |
| | Viável | 0.023069 (11; 99.7; 99.5) | 0.019664 (9; 99.2; 99.5) | **0.006217 (8; 97.9; 98.5)** |
| | PFTrue | 0.017165 (7; 96.1; 99.5) | 0.02387 (6; 94.6; 98.5) | **0.015156 (7; 96.9; 99.5)** |

Tabela B.10: DE para o experimento com 3 objetivos com o algoritmo R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | 0.200166 (3; 78.3; 96.0) | 0.073644 (4; 92.4; 100.0) | **0.062683 (5; 97.1; 100.0)** |
| | Viável | 0.028284 (6; 100.0; 100.0) | 0.065895 (4; 92.4; 100.0) | **0.024850 (5; 98.1; 100.0)** |
| | PFTrue | **0.060773 (4; 92.4; 100.0)** | 0.225382 (3; 75.4; 98.6) | 0.197994 (3; 78.3; 96.0) |
| CAS | Inviável | 0.192508 (4; 78.8; 100.0) | **0.028653 (9; 99.5; 100.0)** | 0.078404 (5; 90.3; 99.4) |
| | Viável | **0.056612 (9; 100.0; 100.0)** | 0.072177 (5; 89.4; 98.9) | 0.121243 (4; 84.1; 98.3) |
| | PFTrue | 0.135317 (4; 84.5; 97.2) | 0.237237 (3; 74.4; 96.1) | **0.012693 (7; 97.3; 100.0)** |
| WS | Inviável | 0.027510 (10; 98.3; 98.9) | **0.023117 (8; 94.9; 97.4)** | **0.023117 (8; 94.9; 97.4)** |
| | Viável | 0.028284 (12; 100.0; 100.0) | **0.020794 (11; 99.4; 99.4)** | 0.028284 (12; 100.0; 100.0) |
| | PFTrue | 0.023727 (12; 100.0; 100.0) | **0.007193 (10; 98.5; 98.9)** | **0.007193 (10; 98.5; 98.9)** |
| E-Shop | Inviável | **0.012695 (10; 99.4; 100.0)** | 0.014280 (11; 99.7; 100.0) | 0.015352 (13; 98.2; 100.0) |
| | Viável | **0.026564 (11; 99.7; 100.0)** | 0.156753 (4; 83.5; 92.0) | 0.028497 (16; 100.0; 100.0) |
| | PFTrue | **0.036564 (12; 100.0; 100.0)** | **0.036564 (12; 100.0; 100.0)** | 0.037298 (16; 100.0; 100.0) |

Considerando o número médio de soluções e porcentagem destas na ROI, para o algoritmo r-NSGA-II-HH com os 3 métodos de seleção (Tabela B.11), é observado que o algoritmo r-NSGA-II-HH-CF gera os menores números na ROI, sendo melhor principalmente na menor e maior instâncias (James e E-shop). Para o algoritmo r-NSGA-II-HH-CF e r-NSGA-II-HH-FRR todas soluções geradas estão 100% dentro da ROI, já o algoritmo r-NSGA-II-HH-R não obteve uma porcentagem ótima em um caso, sendo que para o FM WS e RP real atingiu 99.39%, mesmo com um menor número de soluções na ROI. De maneira geral, o algoritmo r-NSGA-II-HH-FRR gera um maior número de soluções dentro da ROI em todos os FMs e RPs.

Tabela B.11: Número e porcentagem de soluções para o experimento de 3 objetivos com r-NSGA-II-HH.

| FM | RP | r-NSGA-II-HH-CF | r-NSGA-II-HH-FRR | r-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **1.2 (100%)** | **1.2 (100%)** | 1.4 (100%) |
| | Viável | **1.6 (100%)** | 1.7 (100%) | 1.9 (100%) |
| | PFTrue | **1.3 (100%)** | **1.3 (100%)** | 1.6 (100%) |
| CAS | Inviável | **1.4 (100%)** | 1.5 (100%) | 1.6 (100%) |
| | Viável | 1.7 (100%) | 1.5 (100%) | **1.4 (100%)** |
| | PFTrue | 1.6 (100%) | 1.7 (100%) | **1.3 (100%)** |
| WS | Inviável | 5.3 (100%) | **4.9 (100%)** | 5.2 (100%) |
| | Viável | 3.8 (100%) | 2.8 (100%) | **2.7 (100%)** |
| | PFTrue | 3.9 (100%) | 3.7 (100%) | **3.1 (99.39%)** |
| E-Shop | Inviável | **1.9 (100%)** | 2.3 (100%) | 2.5 (100%) |
| | Viável | **2.2 (100%)** | 2.7 (100%) | 2.7 (100%) |
| | PFTrue | **3.0 (100%)** | 3.6 (100%) | **3.0 (100%)** |
| Média | Inviável | 2,4 (100%) | 2,5 (100%) | 2,7 (100%) |
| | Viável | 2,3 (100%) | 2,2 (100%) | 2,2 (100%) |
| | PFTrue | 2,4 (100%) | 2,6 (100%) | 2,2 (99,84%) |

Da mesma forma, a Tabela B.12 mostra o número médio de soluções na ROI e a taxa entre o número de soluções encontradas na ROI e o número total de soluções encontradas, para o algoritmo R-NSGA-II-HH com cada método de seleção. Os valores em negrito representam o menor número de soluções na ROI.

É possível observar na tabela que o algoritmo R-NSGA-II-HH-CF gera os menores números de solução na ROI, sendo melhor principalmente nas menores instâncias (James e CAS). É importante notar que para as maiores instâncias o algoritmo R-NSGA-II-HH-R obteve os menores números de soluções geradas, exceto no RP WS e nos pontos viável e real, com as maiores porcentagens de soluções na ROI, destacando-se ainda, no FM E-Shop onde estes valores foram iguais a 100%. Além disso, o algoritmo R-NSGA-II-HH-FRR foi o que mais gerou em média soluções na ROI em todos os pontos, variando uma porcentagem de soluções entre 79.34% a 88.89%.

Um ponto a se destacar, é o fato de a porcentagem de soluções dentro da ROI aumentar no experimento com 3 objetivos em relação ao experimento com 2 objetivos. Isso é justificado pelo fato do algoritmo R-NSGA-II-HH ter uma maior facilidade em encontrar soluções e uma certa facilidade em manter a diversidade das soluções, caracterizada pela relação de dominância de Pareto, o que não ocorre na relação de r-dominância. Com isso, como a dificuldade aumenta no experimento com 3 objetivos, assim como o número de soluções no espaço de busca, o algoritmo acaba tornando-se mais eficiente para encontrar mais soluções dentro da ROI, mas em contrapartida, também encontra mais soluções fora dela.

Tabela B.12: Número e porcentagem de soluções para o experimento de 3 objetivos com R-NSGA-II-HH.

| FM | RP | R-NSGA-II-HH-CF | R-NSGA-II-HH-FRR | R-NSGA-II-HH-R |
|---|---|---|---|---|
| James | Inviável | **4.9 (85.77%)** | 6.7 (68.99%) | 7.3 (69.90%) |
| | Viável | **2.7 (63.08%)** | 3.6 (37.61%) | 3.7 (34.56%) |
| | Real | **4.4 (81.50%)** | 6.2 (63.78%) | 6.5 (59.31%) |
| CAS | Inviável | **7.9 (90.91%)** | 10.7 (93.04%) | 10.1(94.46%) |
| | Viável | **8.5 (78.43%)** | 10.3 (89.97%) | 9.9 (87.21%) |
| | Real | **9.4 (83.56%)** | 10.7 (89.23%) | 10.5 (91.49%) |
| WS | Inviável | 14.1 (94.39%) | 11.8 (97.57%) | **11.1 (97.58%)** |
| | Viável | **8.3 (84.29%)** | 10.8 (94.34%) | 10.1 (97.68%) |
| | Real | 11.2 (95.28%) | **9.8 (96.94%)** | 11.1 (96.94%) |
| E-Shop | Inviável | 9.6 (96.00%) | 13.0 (95.99%) | **4.4 (100%)** |
| | Viável | 7.7 (89.35%) | 12.2 (95.47%) | **4.2 (100%)** |
| | Real | 8.1 (93.95%) | 13.0 (96.08%) | **4.5 (100%)** |
| Média | Inviável | 9.1 (91.76%) | 10.5 (88.89%) | 9.2 (90.48%) |
| | Viável | 6.8 (78.78%) | 9.2 (79.34%) | 6.9 (79.86%) |
| | Real | 8.2 (88.57%) | 9.9(86.5%) | 8.1 (86.93%) |

De maneira geral, os algoritmos r-NSGA-II-HH-FRR e r-NSGA-II-HH-R foram equivalentes tratando-se das médias de R-HV. O algoritmo r-NSGA-II-HH-R consegue, em geral, encontrar a solução mais próxima ao RP. Em relação ao número de soluções na ROI, há um equilíbrio entre os dois algoritmos, porém o r-NSGA-II-HH-FRR produziu em todos os casos, somente soluções na ROI, o que não ocorre no caso do r-NSGA-II-HH-R. Além disso, o algoritmo r-NSGA-II-HH-CF apresenta, no geral, o menor número de soluções na ROI.

Já para o algoritmo R-NSGA-II-HH com os 3 métodos de seleção, o algoritmo R-NSGA-II-HH-FRR apresenta os melhores resultados, gerando um maior número de soluções dentro da ROI, entretanto com uma menor porcentagens dessas soluções dentro da ROI. Já o algoritmo R-NSGA-II-HH-CF não consegue gerar tantas soluções dentro da ROI, porém ainda se mostra uma boa opção para soluções mais próximas ao RP. Porém, gerou os piores resultados de R-HV.

# APÊNDICE C: QP-3: HH X PEMOAS - RESULTADOS

Neste apêndice são apresentados os resultados obtidos para responder à questão de pesquisa 3 (QP-3).

**Experimento com 2 objetivos**

A Tabela C.1 apresenta os valores médios e o desvio padrão para o indicador R-HV. Os valores em negrito representam os melhores resultados e as células em cinza claro representam os valores estatisticamente equivalentes, levando em consideração o teste estatístico Kruskal-Wallis com um nível de confiança de 95%.

Tabela C.1: Comparação R-HV para 2 objetivos entre HH e PEMOA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | 0.822181 (0.070595) | **0.849487 (0.011108)** | 0.80574 (0.134297) | 0.666078 (0.110133) |
| | Viável | 0.894145 (0.007254) | **0.919008 (0.004812)** | 0.884088 (0.018876) | 0.882451 (0.00691) |
| | Real | 0.890504 (0.012149) | 0.898279 (0.004208) | 0.908972 (0.0239) | **0.926268 (0.009816)** |
| Cas | Inviável | 0.903875 (0.040236) | **0.937531 (0.024251)** | 0.927352 (0.063159) | 0.900968 (0.07969) |
| | Viável | 0.947406 (0.003698) | 0.954228 (0.008669) | 0.962304 (0.020654) | **0.97387 (0.009275)** |
| | Real | 0.945245 (0.006969) | 0.952888 (0.005434) | 0.965468 (0.012508) | **0.975569 (0.004943)** |
| WS | Inviável | 0.903808 (0.026074) | **0.953677 (0.005914)** | 0.942189 (0.020742) | 0.929605 (0.051043) |
| | Viável | 0.944466 (0.002482) | 0.953761 (0.007617) | 0.959501 (0.012877) | **0.967229 (0.009898)** |
| | Real | 0.95918 (0.003012) | 0.962274 (8.87E-4) | 0.961487 (0.011994) | **0.979089 (0.002602)** |
| E-Shop | Inviável | 0.946005 (0.029347) | 0.961268 (0.009387) | **0.964527 (0.013282)** | 0.951833 (0.058212) |
| | Viável | 0.945853 (0.004724) | 0.955645 (0.007969) | 0.955341 (0.01486) | **0.966083 (0.012696)** |
| | Real | 0.968745 (0.006675) | 0.977753 (0.003957) | 0.9665 (0.027584) | **0.9869 (0.004791)** |

De maneira geral, é possível observar na tabela uma prevalência de melhores resultados para o algoritmo R-NSGA-II-HH-FRR. Entretanto, na grande o maioria dos casos sem diferença estatística. Além disso, o algoritmo R-NSGA-II-HH-FRR tende a ser uma melhor opção nos casos de RP reais e também quando o número de produtos que podem ser derivados pela LPS aumenta. O algoritmo r-NSGA-II, foi equivalente ao melhor resultado apenas no FM James e no RP inviável.

Analisando cada PEMOA com sua respectiva versão HH, pode-se observar que o r-NSGA-II-HH-FRR apresenta melhores resultados de R-HV em relação ao r-NSGA-II, em todos os FMs e RPs. O algoritmo R-NSGA-II-HH-FRR consegue ser melhor na grande maioria dos casos, mas sem diferença estatística. É importante notar que nas instâncias maiores, o algoritmo R-NSGA-II-FRR tende a apresentar resultados melhores, sem a ocorrência de equivalência estatista com o R-NSGA-II.

Considere agora a solução mais próxima dos RPs gerados pelos algoritmos levando em consideração todas as execuções e removendo as soluções repetidas (conjunto $PF_{known}$). A Tabela C.2 apresenta os valores de DE entre a solução e o RP, e os valores de *fitness* da solução selecionada (a que está mais próxima). Os valores em negrito representam os melhores resultados.

Tabela C.2: Comparação DE para 2 objetivos entre HH e PEMOA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | 0.19752 (3;78.3) | **0.06278 (4; 92.4)** | 0.19752 (3;78.3) | 0.19752 (3; 78.3) |
| | Viável | 0.04056 (6;99.0) | **0.03443 (5; 98.1)** | 0.05000 (6;100.0) | 0.05000 (6; 100.0) |
| | Real | **0.01113 (5;98.1)** | **0.01113 (5; 98.1)** | 0.01113 (5;98.1) | 0.01113 (5; 98.1) |
| CAS | Inviável | **0.06266 (5;90.7)** | 0.07585 (5; 89.4) | 0.10225 (5;86.7) | 0.11980 (4; 85.0) |
| | Viável | 0.03559 (8;99.5) | 0.02248 (7; 98.2) | **0.00525 (7;96.4)** | 0.00942 (7; 96.9) |
| | Real | **0.00083 (7;96.9)** | 0.01693 (8; 98.6) | 0.00356 (7;97.3) | **0.00083 (7; 96.9)** |
| WS | Inviável | 0.04355 (7;92.7) | **0.01055 (8; 96.6)** | 0.04907 (7;92.1) | 0.04631 (7; 92.4) |
| | Viável | 0.02327 (10;98.3) | 0.02077 (9; 98.0) | 0.04004 (12;100.0) | **0.01803 (9; 97.7)** |
| | Real | 0.00599 (10;98.5) | 0.00879 (10; 98.8) | **0.00240 (10;97.7)** | 0.00879 (10; 98.8) |
| E-Shop | Inviável | 0.01100 (9;97.9) | 0.02092 (6; 94.9) | **0.00782 (12;96.9)** | 0.09438 (5; 87.5) |
| | Viável | 0.03735 (9;98.7) | **0.02486 (7; 97.4)** | 0.02974 (13;97.9) | 0.04239 (10; 99.2) |
| | Real | 0.01049 (8;96.9) | **0.00736 (8; 98.7)** | 0.03087 (12;94.9) | 0.01494 (10; 99.4) |

Na tabela é possível notar um maior número de melhores resultados para o algoritmo r-NSGA-II-HH-FRR, sendo melhor na metade dos casos (6 de 12). Já o algoritmo R-NSGA-II-HH-FRR obtém o melhor resultado em dois únicos casos. Os PEMOAs obtiveram o mesmo número de melhores soluções neste experimento (3 de 12). Observa-se a mesma relação entre os PEMOAs e suas versões HHs descrita anteriormente, ou seja, o algoritmo HH consegue melhorar a versão tradicional.

A Tabela C.3 apresenta o número médio de soluções na ROI e a taxa entre o número de soluções encontradas na ROI e o número total de soluções encontradas. Os valores em negrito representam o menor número de soluções na ROI.

Assim como já ocorreu neste indicador de qualidade anteriormente, o algoritmo r-NSGA-II apresenta o menor número médio de soluções na ROI em todos os casos. Além disso, observa-se que o algoritmo r-NSGA-II-HH-FRR apresenta mais soluções na ROI em relação ao r-NSGA-II com uma porcentagem maior de soluções na ROI variando entre 99.44% a 100%. Da mesma forma, os algoritmos R-NSGA-II e R-NSGA-II-HH-FRR apresentaram as maiores médias de soluções dentro da ROI, sendo que o segundo, apresenta resultados ainda superiores, porém, com porcentagens de soluções na ROI menores, variando entre 49.2% a 73.1%.

Resumindo, o algoritmo r-NSGA-II-HH-FRR consegue ser melhor que o algoritmo r-NSGA-II na grande maioria dos casos, mesmo gerando um número superior de soluções na ROI, obteve maiores porcentagens dessas, ou seja, quase não produz soluções fora da ROI em todos os casos. Em relação aos algoritmos R-NSGA-II e R-NSGA-II-HH-FRR, o algoritmo HH consegue obter um maior número de soluções na ROI porém, também produz uma maior quantidade de soluções fora da ROI. Além disso, o R-NSGA-II consegue produzir soluções mais próximas ao ponto de referência em relação ao R-NSGA-II-HH-FRR.

De modo geral, o algoritmo r-NSGA-II-HH-FRR mostra-se uma boa opção quando o testador deseja apenas uma solução especifica, tendo em vista que consegue produzir um menor número de soluções, e quase sempre, somente soluções na ROI, sendo ainda, as que mais se aproximam do RP. Já o algoritmo R-NSGA-II-HH-FRR apresenta as soluções mais diversificadas

Tabela C.3: Número e porcentagem de soluções para o experimento de 2 objetivos entre HH e PEMOA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | **1.0 (100%)** | **1.0 (100%)** | 1.6 (54.11%) | 1.4 (19.98%) |
| | Viável | **1.0 (100%)** | 2.0 (100%) | 2.3 (56.83%) | 3.0 (33.78%) |
| | Real | **1.0 (100%)** | **1.0 (100%)** | 2.3 (65.38%) | 3.2 (41.71%) |
| CAS | Inviável | **1.7 (92.77%)** | 2.0 (97.77%) | 3.0 (71.61%) | 5.0 (52.47%) |
| | Viável | **1.0 (100%)** | 1.6 (100%) | 3.9 (81.38%) | 5.1 (85.67%) |
| | Real | 1.3 (100%) | **1.1 (100%)** | 4.2 (83.96%) | 5.6 (80.87%) |
| WS | Inviável | **1.6 (100%)** | 1.8 (100%) | 3.1 (87.71%) | 6.1 (59.62%) |
| | Viável | **1.1 (100%)** | 2.1 (100%) | 3.8 (95.27%) | 5.3 (89.75%) |
| | Real | **1.0 (100%)** | 1.7 (100%) | 3.4 (97.52%) | 6.0 (76.71%) |
| E-Shop | Inviável | **2.0 (92.77%)** | 2.1 (100%) | 3.8 (98.00%) | 6.6 (64.73%) |
| | Viável | **1.2 (100%)** | 2.5 (100%) | 4.1 (98.66%) | 6.9 (83.16%) |
| | Real | **1.9 (100%)** | 2.2 (100%) | 4.4 (93.61%) | 6.9 (85.10%) |
| Média | Inviável | 1.6 (96.38%) | 1.7 (99.44%) | 2.8 (77.85%) | 4.7 (49.2%) |
| | Viável | 1.1 (100%) | 2.1 (100%) | 3.5 (83.03%) | 5.1 (73.1%) |
| | Real | 1.3 (100%) | 1.5 (100%) | 3.5 (85.11%) | 5.4 (71.1%) |

na ROI, entretanto, relativamente mais distantes do que os outros algoritmos em relação ao RP. Além disso, o R-NSGA-II ainda pode ser considerado uma melhor opção quando o testador deseja um melhor *trade-off* na ROI, pois consegue ser mais eficiente em relação ao algoritmo R-NSGA-II-HH-FRR na geração de soluções somente na ROI.

Ambos os algoritmos HHs conseguem obter resultados melhores que os respectivos PEMOAs. Entretanto, há uma melhora significativa para o algoritmo r-NSGA-II, o que não acontece no caso do algoritmo R-NSGA-II. Uma possível explicação é que o R-NSGA-II-HH não aplica o conceito de r-dominância, sendo ainda igual a HH tradicional, por isso, talvez, não tenha conseguido melhorar, com grande diferença, o desempenho do correspondente R-NSGA-II.

### Experimento com 3 objetivos

A Tabela C.4 apresenta os valores médios e o desvio padrão para o indicador R-HV. De maneira geral, pode-se notar que o algoritmo R-NSGA-II-HH-FRR obtém os melhores resultados ou estatisticamente equivalentes ao melhor em quase todos os casos, exceto no FM CAS e RP inviável, onde não obteve equivalência estatística ao melhor. Deve-se destacar que o algoritmo R-NSGA-II apresentou os resultados levemente maiores na maior instância (E-Shop). Assim como no experimento com 2 objetivos, o algoritmo r-NSGA-II não obteve bons resultados de R-HV em relação aos demais algoritmos.

A Tabela C.5 mostra os resultados do indicador DE nesta comparação. Assim como no experimento com 2 objetivos, o algoritmo r-NSGA-II-HH-FRR obtém um maior número de soluções mais próximas ao ponto de referência em relação aos demais algoritmos (6 de 12). Além disso, é possível notar que mesmo gerando uma única melhor solução em todo o escopo do experimento, o algoritmo R-NSGA-II ainda produz soluções melhores do que o algoritmo R-NSGA-II-HH-FRR, em quase todos os casos. Este fato, também ocorre quando comparados os algoritmo r-NSGA-II e r-NSGA-II-HH-FRR.

Quanto ao número de soluções encontradas, a Tabela C.6 mostra que o algoritmo r-NSGA-II-HH-FRR apresenta o menor valor médio de soluções na ROI nos RPs inviáveis e viáveis. Já para RPs reais o algoritmo r-NSGA-II apresenta o menor valor. Além disso, ambos os

Tabela C.4: Comparação R-HV para 3 objetivos entre HH e PEMOA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | 0.573699 (0.136175) | **0.789351 (0.011961)** | 0.724055 (0.155819) | 0.690575 (0.110315) |
| | Viável | 0.859696 (0.057611) | 0.861724 (0.080063) | 0.885777 (0.035451) | **0.909189 (0.007094)** |
| | Real | 0.770768 (0.044958) | 0.788002 (0.031475) | 0.817249 (0.032007) | **0.834227 (0.015283)** |
| CAS | Inviável | 0.831569 (0.043318) | **0.91302 (0.018707)** | 0.816171 (0.086145) | 0.853895 (0.049637) |
| | Viável | 0.889719 (0.032903) | 0.884443 (0.039629) | **0.908471 (0.055839)** | 0.900703 (0.063162) |
| | Real | 0.895692 (0.041196) | **0.92782 (0.026705)** | 0.886272 (0.058637) | 0.908511 (0.038665) |
| WS | Inviável | 0.874067 (0.032006) | 0.919196 (0.021416) | 0.917037 (0.033608) | **0.940178 (0.04266)** |
| | Viável | 0.942668 (0.007381) | 0.95642 (0.009214) | 0.954994 (0.015334) | **0.961598 (0.011756)** |
| | Real | 0.94076 (0.007188) | 0.943381 (0.008104) | 0.954382 (0.016987) | **0.969412 (0.006575)** |
| E-Shop | Inviável | 0.946127 (0.01389) | 0.956346 (0.014745) | **0.95677 (0.021565)** | 0.949669 (0.038142) |
| | Viável | 0.935751 (0.015254) | 0.961349 (0.011311) | **0.969794 (0.011491)** | 0.966667 (0.011692) |
| | Real | 0.932253 (0.013813) | 0.923502 (0.025139) | **0.960462 (0.022583)** | 0.958038 (0.024681) |

Tabela C.5: Comparação DE para 3 objetivos entre HH e PMOEA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | 0.20016 (3;78;96) | **0,07364 (4; 92,4; 100,0)** | 0.20016 (3;78;96) | **0,07364 (4; 92,4; 100,0)** |
| | Viável | **0.02485 (5;98;100)** | 0,06314 (4; 92,4; 98,6) | 0.02828 (6;100;100) | 0,06589 (4; 92,4; 100,0) |
| | PFTrue | **0.06077 (4;92;100)** | **0,06077 (4; 92,4; 100,0)** | **0.06077 (4;92;100)** | 0,22538 (3; 75,4; 98,6) |
| CAS | Inviável | 0.03264 (6;95;99) | 0,03893 (6; 94,2; 98,9) | 0.18762 (4;79;99) | **0,02865 (9; 99,5; 100,0)** |
| | Viável | 0.04229 (7;97;100) | **0,03579 (7; 96,9; 99,4)** | 0.05661 (9;100;100) | 0,07217 (5; 89,4; 98,9) |
| | PFTrue | 0.04720 (6;93;100) | **0,02249 (7; 96,0; 100,0)** | 0.08630 (5;89;100) | 0,23723 (3; 74,4; 96,1) |
| WS | Inviável | 0.05573 (7;92;94) | **0,01476 (8; 96,0; 96,4)** | 0.02345 (8;94;96) | 0,02311 (8; 94,9; 97,4) |
| | Viável | **0.01210 (10;98;98)** | 0,02079 (11; 99,4; 99,4) | 0.02828 (12;100;100) | 0,02079 (11; 99,4; 99,4) |
| | PFTrue | 0.00985 (9;97;98) | 0,01594 (9; 97,7; 97,4) | 0.07464 (7;92;94) | **0,00719 (10; 98,5; 98,9)** |
| E-Shop | Inviável | 0.02003 (23;98;100) | 0,02576 (7; 96,4; 99,0) | 0.07491 (10;93;94) | **0,01428 (11; 99,7; 100,0)** |
| | Viável | **0.00987 (20;98;98)** | 0,01966 (9; 99,2; 99,5) | 0.04208 (10;93;97) | 0,15675 (4; 83,5; 92,0) |
| | PFTrue | 0.03723 (16;93;98) | **0,0238 (6; 94,6; 98,5)** | 0.03289 (14;99;100) | 0,03656 (12; 100,0; 100,0) |

algoritmos obtêm em todos os casos 100% de soluções na ROI. O algoritmo R-NSGA-II-HH-FRR apresenta o maior valor médio de soluções na ROI, em todos os casos com uma porcentagem de soluções entre 95,47% a 96.08%. O mesmo ocorre com o algoritmo R-NSGA-II, ele é capaz de gerar mais soluções na ROI em relação ao r-NSGA-II e r-NSGA-II-HH-FRR, porém gera um menor número de soluções que o R-NSGA-II-HH-FRR, mas possui uma porcentagem de soluções superior, variando entre 97.43% a 98.82%.

De maneira geral, os resultados obtidos no experimento com 3 objetivos, assemelharam-se aos resultados obtidos pelo experimento com 2 objetivos. Sendo assim, o algoritmo r-NSGA-II-HH-FRR mostra-se uma boa opção quando o testador deseja, apenas uma solução especifica, tendo em vista que consegue produzir um menor número de soluções, quase sempre, somente soluções na ROI, e estas soluções ainda são as que mais se aproximam do RP. Já o algoritmo R-NSGA-II-HH-FRR apresenta as soluções mais diversificadas na ROI, entretanto, relativamente mais distantes do que os outros algoritmos em relação ao RP. Além disso, o R-NSGA-II ainda pode ser considerado uma melhor opção quando o testador deseja um melhor *trade-off* na ROI, pois consegue ser mais eficiente em relação ao algoritmo R-NSGA-II-HH-FRR na geração de soluções somente na ROI.

Tabela C.6: Número e porcentagem de soluções para o experimento de 3 objetivos entre HH e PEMOA.

| FM | RP | r-NSGA-II | r-NSGA-II-HH-FRR | R-NSGA-II | R-NSGA-II-HH-FRR |
|---|---|---|---|---|---|
| James | Inviável | 1.8 (100%) | **1,2 (100%)** | 2.4 (81.83%) | 6,7 (68,99%) |
| | Viável | **1.0 (100%)** | 1,7 (100%) | 2.5 (85.44%) | 3,6 (37,61%) |
| | Real | **1.3 (100%)** | **1,3 (100%)** | 2.7 (89.78%) | 6,2 (63,78%) |
| CAS | Inviável | 2.2 (100%) | **1,5 (100%)** | 8.1 (90.56%) | 10,7 (93,04%) |
| | Viável | **1.3 (100%)** | 1,5 (100%) | 6.5 (90.56%) | 10,3 (89,97%) |
| | Real | 2.0 (100%) | **1,7 (100%)** | 7.3 (86.24%) | 10,7 (89,23%) |
| WS | Inviável | **3.4 (100%)** | 4,9 (100%) | 3.7 (100%) | 11,8 (97,57%) |
| | Viável | **1.2 (100%)** | 2,8 (100%) | 3.9 (99.39%) | 10,8 (94,34%) |
| | Real | **2.0 (100%)** | 3,7 (100%) | 3.6 (99.17%) | 9,8 (96,94%) |
| E-Shop | Inviável | 3.1 (100%) | **2,3 (100%)** | 7.8 (98.82%) | 13,0 (95,99%) |
| | Viável | 3.0 (100%) | **2,7 (100%)** | 7.8 (98.77%) | 12,2 (95,47%) |
| | Real | **2.9 (100%)** | 3,6 (100%) | 7.7 (97.43%) | 13,0 (96,08%) |
| Média | Inviável | 2.1 (100%) | 2,5 (100%) | 5.5 (92.8%) | 10.5 (88.89%) |
| | Viável | 1.6 (100%) | 2,2 (100%) | 5.1 (93.5%) | 9.2 (79.34%) |
| | Real | 2.0 (100%) | 2,6 (100%) | 5.3 (93.1%) | 9.9(86.5%) |

# APÊNDICE D: QP-4: R-NSGA-II-HH E R-NSGA-II-HH X NSGA-II-HH - RESULTADOS

Neste apêndice são apresentados os resultados obtidos para responder à questão de pesquisa 4 (QP-4).

### Experimentos com 2 objetivos

A Tabela D.1 apresenta as médias e desvio padrão dos valores de R-HV, para os três algoritmos. É possível observar na tabela que o algoritmo R-NSGA-II-HH-FRR apresenta os melhores resultados ou estatisticamente equivalentes ao melhor na grande maioria dos FMs e RPS, exceto apenas, no FM James e nos RPs inviável e viável. O algoritmo r-NSGA-II-HH-FRR apresenta o melhor resultado em todos os FMs quando o RP informado é inviável. Já o algoritmo NSGA-II-HH apresenta resultados inferiores aos demais algoritmos, obtendo melhor resultado ou equivalente ao melhor, nos RPs viáveis nas instâncias CAS, WS e E-Shop.

Tabela D.1: Resultados de R-HV para (r e R)-NSGA-II-HH e NSGA-II-HH, considerando a formulação de 2 objetivos.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 0.75087 (0.0) | **0.849487 (0.011108)** | 0.666078 (0.110133) |
| | Viável | 0.91264 (0.0) | **0.919008 (0.004812)** | 0.882451 (0.00691) |
| | Real | 0.90497 (0.00436) | 0.898279 (0.004208) | **0.926268 (0.009816)** |
| CAS | Inviável | 0.84089 (0.00678) | **0.937531 (0.024251)** | 0.900968 (0.07969) |
| | Viável | 0.97354 (0.00573) | 0.954228 (0.008669) | **0.97387 (0.009275)** |
| | Real | 0.95482 (0.0045) | 0.952888 (0.005434) | **0.975569 (0.004943)** |
| WS | Inviável | 0.82742 (0.00259) | **0.953677 (0.005914)** | 0.929605 (0.051043) |
| | Viável | **0.97097 (0.00201)** | 0.953761 (0.007617) | 0.967229 (0.009898) |
| | Real | 0.96504 (0.00262) | 0.962274 (8.87E-4) | **0.979089 (0.002602)** |
| E-Shop | Inviável | 0.87451 (0.02261) | **0.961268 (0.009387)** | 0.951833 (0.058212) |
| | Viável | **0.99065 (0.00144)** | 0.955645 (0.007969) | 0.966083 (0.012696) |
| | Real | 0.98246 (0.0032) | 0.977753 (0.003957) | **0.9869 (0.004791)** |

Avaliando a solução mais próxima ao RP, a Tabela D.2 apresenta os resultados de DE para os algoritmos. Observa-se que o algoritmo r-NSGA-II-HH-FRR apresenta as soluções mais próximas ao RP na grande maioria dos casos (9 de 12), sendo melhor principalmente no menor e maior FMs, James e E-Shop, respectivamente. O algoritmo NSGA-II-HH obtém o melhor resultado em apenas dois casos (2 de 12), nos FMs James e WS, para o RP real.

Tabela D.2: Resultados de DE para (r e R)-NSGA-II-HH e NSGA-II-HH, considerando a formulação de 2 objetivos.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 0,19752 (3; 78,3) | **0.06278 (4; 92.4)** | 0.19752 (3; 78.3) |
| | Viável | 0,05000 (6; 100) | **0.03443 (5; 98.1)** | 0.05000 (6; 100.0) |
| | Real | **0,01113 (5; 98,1)** | **0.01113 (5; 98.1)** | **0.01113 (5; 98.1)** |
| CAS | Inviável | 0,11099 (4; 85,9) | **0.07585 (5; 89.4)** | 0.11980 (4; 85.0) |
| | Viável | 0,04006 (9; 100) | 0.02248 (7; 98.2) | **0.00942 (7; 96.9)** |
| | Real | 0,00797 (7; 97,7) | 0.01693 (8; 98.6) | **0.00083 (7; 96.9)** |
| WS | Inviável | 0,13252 (5; 83,7) | **0.01055 (8; 96.6)** | 0.04631 (7; 92.4) |
| | Viável | 0,03439 (11; 99,4) | 0.02077 (9; 98.0) | **0.01803 (9; 97.7)** |
| | Real | **0,00879 (10; 98,8)** | **0.00879 (10; 98.8)** | **0.00879 (10; 98.8)** |
| E-Shop | Inviável | 0,12228 (4; 84,7) | **0.02092 (6; 94.9)** | 0.09438 (5; 87.5) |
| | Viável | 0,05000 (12; 100) | **0.02486 (7; 97.4)** | 0.04239 (10; 99.2) |
| | Real | 0,01494 (10; 99,4) | **0.00736 (8; 98.7)** | 0.01494 (10; 99.4) |

Por fim, analisado a quantidade de soluções na ROI e a porcentagem de soluções geradas que estão na ROI. A Tabela D.3 apresenta que o algoritmo r-NSGA-II-HH-FRR obtém as médias de menores valores dentro da ROI e ainda obtém as melhores porcentagens de soluções na ROI. O algoritmo R-NSGA-II-HH-FRR mostra-se o oposto, gerando mais soluções na ROI, porém com uma porcentagem menor. Já o algoritmo NSGA-II-HH gerou as menores porcentagens de soluções na ROI, variando entre 12.3% a 48%. Dessa forma, o algoritmo NSGA-II-HH gera um grande número de soluções fora da ROI, o que dificulta para o testador determinar qual solução melhor satisfaz a suas preferências.

Tabela D.3: Número e porcentagem de soluções para o experimento de 3 objetivos entre HH e PEMOA.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 1,0 (16,6%) | **1.0 (100%)** | 1.4 (19.98%) |
| | Viável | 2,0 (33,3%) | **2.0 (100%)** | 3.0 (33.78%) |
| | Real | 2,5 (42,7%) | **1.0 (100%)** | 3.2 (41.71%) |
| CAS | Inviável | **1,1 (11,6%)** | 2.0 (97.77%) | 5.0 (52.47%) |
| | Viável | 4,5 (46,2%) | **1.6 (100%)** | 5.1 (85.67%) |
| | Real | 4,8 (48,9%) | **1.1 (100%)** | 5.6 (80.87%) |
| WS | Inviável | **1,1 (9,4%)** | 1.8 (100%) | 6.1 (59.62%) |
| | Viável | 5,3 (43,2%) | **2.1 (100%)** | 5.3 (89.75%) |
| | Real | 5,3 (43,2%) | **1.7 (100%)** | 6.0 (76.71%) |
| E-Shop | Inviável | **1,1 (11,6%)** | 2.1 (100%) | 6.6 (64.73%) |
| | Viável | 6,5 (53,2%) | **2.5 (100%)** | 6.9 (83.16%) |
| | Real | 7,0 (57,3%) | **2.2 (100%)** | 6.9 (85.10%) |
| Média | Inviável | 1,1 (12,3%) | 1.7 (99.44%) | 4.7 (49.2%) |
| | Viável | 4,6 (43,9%) | 2.1 (100%) | 5.1 (73.1%) |
| | Real | 4,9 (48%) | 1.5 (100%) | 5.4 (71.1%) |

Resumindo os resultados, a abordagem proposta, na maioria dos casos, gera os melhores *trade-offs* em relação ao R-HV, soluções mais próximas dos RPs e um maior e menor número de soluções dentro da ROI. Dessa forma, melhor satisfaz as preferências do testador em relação

ao algoritmo NSGA-II-HH. Além disso, observa-se que o NSGA-II-HH gera quase 83% das soluções fora da ROI para os RPs inviáveis, e para os demais RPs cerca de 52%, tais valores talvez não sejam interessantes do ponto de vista do testador.

### Experimento com 3 objetivos

A Tabela D.4 apresenta os valores médios e o desvio padrão para o indicador R-HV. Assim como na formulação com 2 objetivos, as HH baseadas em preferência obtêm resultados melhores ou estatisticamente equivalentes ao melhor na grande maioria dos FMs e RPs, em relação ao algoritmo NSGA-II-HH. De maneira geral o algoritmo R-NSGA-II-HH-FRR gera os melhores resultados (8 de 12), destacando-se na instâncias maiores, WS e E-Shop.

Tabela D.4: Resultados de R-HV para (r e R)-NSGA-II-HH e NSGA-II-HH, considerando a formulação de 3 objetivos.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 0.74049 (0.0) | **0.789351 (0.011961)** | 0.690575 (0.110315) |
| | Viável | **0.91264 (7.0E-5)** | 0.861724 (0.080063) | 0.909189 (0.007094) |
| | Real | **0.86543 (0.00934)** | 0.788002 (0.031475) | 0.834227 (0.015283) |
| CAS | Inviável | 0.88145 (0.04705) | **0.91302 (0.018707)** | 0.853895 (0.049637) |
| | Viável | **0.95978 (0.02052)** | 0.884443 (0.039629) | 0.900703 (0.063162) |
| | Real | 0.90403 (0.02041) | **0.92782 (0.026705)** | 0.908511 (0.038665) |
| WS | Inviável | 0.83478 (0.05777) | 0.919196 (0.021416) | **0.940178 (0.04266)** |
| | Viável | **0.9732 (0.00474)** | 0.95642 (0.009214) | 0.961598 (0.011756) |
| | Real | 0.9026 (0.01081) | 0.943381 (0.008104) | **0.969412 (0.006575)** |
| E-Shop | Inviável | 0.94666 (0.05075) | **0.956346 (0.014745)** | 0.949669 (0.038142) |
| | Viável | 0.92297 (0.01439) | 0.961349 (0.011311) | **0.966667 (0.011692)** |
| | Real | 0.90944 (0.02511) | 0.923502 (0.025139) | **0.958038 (0.024681)** |

Considerando o indicador de qualidade DE (Tabela D.5), observam-se resultados semelhantes ao ocorrido na formulação com 2 objetivos. O algoritmo r-NSGA-II-HH-FRR apresenta os melhores resultados em 8 de 12 casos. O algoritmo r-NSGA-II-HH-FRR obtém 5 melhores resultados de 12. Já o algoritmo NSGA-II-HH consegue a solução mais próxima ao RP em apenas dois casos (2 de 12), sendo no FM James e nos RPs viável e real.

Novamente, analisado o número de soluções dentro da ROI, na Tabela D.6 observa-se que para todos os casos, o algoritmo r-NSGA-II-HH-FRR apresentou 100% de soluções na ROI. O algoritmo R-NSGA-II-HH-FRR foi o que mais gerou soluções na ROI, com uma porcentagem média entre 79.34% a 88.89%. O algoritmo NSGA-II-HH fica em um ponto médio entre os outros dois algoritmos na quantidade de soluções na ROI, entretanto, a porcentagem de solução nesta é bem inferior, variando em média de 30.3% a 32.9%.

De maneira geral, os resultados obtidos na formulação com 3 objetivos assemelham-se aos resultados encontrados na formulação com 2 objetivos. Novamente, as HH baseadas em preferência obtêm melhores resultados na grande maioria dos casos e experimentos em relação ao algoritmo NSGA-II-HH.

Tabela D.5: Resultados de DE para (r e R)-NSGA-II-HH e NSGA-II-HH, considerando a formulação de 3 objetivos.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 0,22729 (3; 75,4; 98,6) | **0,07364 (4; 92,4; 100,0)** | **0,07364 (4; 92,4; 100,0)** |
| | Viável | **0,02828 (6; 100; 100)** | 0,06314 (4; 92,4; 98,6) | 0,06589 (4; 92,4; 100,0) |
| | PFTrue | **0,06077 (4; 92,4; 100)** | **0,06077 (4; 92,4; 100,0)** | 0,22538 (3; 75,4; 98,6) |
| CAS | Inviável | 0,16128 (4; 81,9; 99,4) | 0,03893 (6; 94,2; 98,9) | **0,02865 (9; 99,5; 100,0)** |
| | Viável | 0,05661 (9; 100; 100) | **0,03579 (7; 96,9; 99,4)** | 0,07217 (5; 89,4; 98,9) |
| | PFTrue | 0,07705 (5; 90,3; 99,4) | **0,02249 (7; 96,0; 100,0)** | 0,23723 (3; 74,4; 96,1) |
| WS | Inviável | 0,16129 (5; 82,3; 90,2) | **0,01476 (8; 96,0; 96,4)** | 0,02311 (8; 94,9; 97,4) |
| | Viável | 0,02828 (12; 100; 100) | **0,02079 (11; 99,4; 99,4)** | **0,02079 (11; 99,4; 99,4)** |
| | PFTrue | 0,01190 (9; 97,4; 97,9) | 0,01594 (9; 97,7; 97,4) | **0,00719 (10; 98,5; 98,9)** |
| E-Shop | Inviável | 0,14441 (5; 84,7; 96,5) | 0,02576 (7; 96,4; 99,0) | **0,01428 (11; 99,7; 100,0)** |
| | Viável | 0,02828 (12; 100; 100) | **0,01966 (9; 99,2; 99,5)** | 0,15675 (4; 83,5; 92,0) |
| | PFTrue | 0,03439 (6; 93,9; 99,5) | **0,0238 (6; 94,6; 98,5)** | 0,03656 (12; 100,0; 100,0) |

Tabela D.6: Número e porcentagem de soluções para o experimento de 3 objetivos entre HH e PEMOA.

| FM | RP | NSGA-II-HH | r-NSGA-II-HH-FRR | R-NSGA-II-HH-FRR |
|---|---|---|---|---|
| James | Inviável | 3 (29,2%) | **1,2 (100%)** | 6,7 (68,99%) |
| | Viável | 2 (19,5%) | **1,7 (100%)** | 3,6 (37,61%) |
| | Real | 3 (14,8%) | **1,3 (100%)** | 6,2 (63,78%) |
| CAS | Inviável | 5 (21,3%) | **1,5 (100%)** | 10,7 (93,04%) |
| | Viável | 5,8 (26,5%) | **1,5 (100%)** | 10,3 (89,97%) |
| | Real | 5,7 (26,5%) | **1,7 (100%)** | 10,7 (89,23%) |
| WS | Inviável | 5,1 (23,0%) | **4,9 (100%)** | 11,8 (97,57%) |
| | Viável | 7,4 (27,1%) | **2,8 (100%)** | 10,8 (94,34%) |
| | Real | 10,2 (38,3%) | **3,7 (100%)** | 9,8 (96,94%) |
| E-Shop | Inviável | 9,3 (47,9%) | **2,3 (100%)** | 13,0 (95,99%) |
| | Viável | 9,9 (58,8%) | **2,7 (100%)** | 12,2 (95,47%) |
| | Real | 9,4 (50,2%) | **3,6 (100%)** | 13,0 (96,08%) |
| Média | Inviável | 5,5 (30,3%) | 2,5 (100%) | 10.5 (88.89%) |
| | Viável | 6,2 (32,9%) | 2,2 (100%) | 9.2 (79.34%) |
| | Real | 7 (32,4%) | 2,6 (100%) | 9.9(86.5%) |